Montanuniversitaet Leoben

Distributed Multi-sensor Fusion System for Drilling Rig State Detection

Doctoral Thesis

Author:
Mohammad Arghad Arnaout

Supervisors:
O.Univ.Prof. Dipl.-Ing. Dr.-techn. Paul O'Leary B.A.,B.A.I.,M.E.E.
Univ.-Prof. Dipl.-Ing. Dr.-mont. Gerhard Thonhauser

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Institute for Automation, Montanuniversitaet Leoben

Declaration of Authorship

Affidavit

I, Mohammad Arghad Arnaout, declare in lieu of oath, that I wrote this thesis and performed the associated research myself, using only literature cited in this volume.

Signed: _____

Date: _____

Acknowledgements

I am honored to appreciate the following people for their invaluable help during the course of my PhD work:

- I would like to express my gratitude to my supervisor Prof. Paul O'Leary for his support and comments on this work. It is a great memory in my life;
- I would also like to thank Prof. Gerhard Thonhauser for his valuable advice and support in the domain of Drilling Engineering;
- I appreciate the valuable input and support from Prof. Klaus Schmaranz in the domain of Information Technology;
- I would like to thank my friend and colleague Bilal Esmael for spending time to discuss the different topics in my thesis;
- Thanks to my family for their continued support and patience over the days that I spent away from them whilst busy with my research, my special thanks to my wife Lara;
- Thanks to my father and my mother who never stop praying for me.

Abstract

This thesis presents a framework for the automatic identification of the state of an oil drilling system from sensor data. The reliable detection of states is a prerequisite for the identification of operations. Although the framework has been developed for monitoring drilling, it is generally applicable to data fusion models for the generation of features and decision making.

The system identifies specific states of the equipment and/or process dependent on predefined sensor information extracted dynamically from the sensor data.

Three fundamental types of states are defined: Cluster States, Trend States, and Shape States. Cluster States are defined by discriminating data into clusters using "Expectation Maximization", "Envelope" and "Otsu" algorithms. Trend States are detected in sensor measurements by applying Piecewise Linear Approximation algorithm where the final trend states are determined after a number of merging operations on small trend sections in data. Shape States are identified in sensor data through the orthonormal polynomials method where the polynomial coefficients are used as shape descriptor for the template shape states.

A distributed state recognition system has been implemented as an embodiment of the proposed framework and as a tool of verifying the proposed methods. Specific subsystems of a drilling rig have been used as example systems whose states can be identified. The sub-systems are: Circulation Sub-system, Rotary Sub-system, and Hoisting Subsystem. The verification process of the recognized states is automatically performed and verified against manually classified states from experts. It is proposed to apply the framework and the concept to analyze the drilling rig performance and optimize the drilling process.

Zusammenfassung

Diese Arbeit praesentiert ein Framework fuer die automatische Identifikation des Zustands eines Bohrsystems einer Erdoelbohrung aus den Sensordaten. Die zuverlaessige Erkennung eines Zustands ist die Vorraussetzung zur Identifikation einer Operation. Obwohl das Framework fr die Ueberwachung von Tiefbohrungen entwickelt wurde, ist es generell fuer die Generierung von features und fuer die Entscheidungsfindung des passenden Datenfusionsmodells anwendbar.

Das System identifiziert abhaengig von zuvor definierten Sensor Informationen - dynamisch aus den Sensordaten die spezifischen Zustaende der Bohrausruestung und / oder der Bohrprozesse. Drei fundamentale Typen von Zustaenden sind definiert: “Cluster States, Trend States und Shape States”. Diese werden durch die Differenzierung der Daten in Gruppen durch “Expectation Maximization”, “Envelope” und “Otsu” Algorithmen definiert. “Trend States” werden in Sensor Messungen durch die Anwendung von stueckweisen liniearen Annaeherungs Algorithmen indem die finalen Trends erst nach der Zusammenfassung einer Anzahl von kleinen Trend Sektionen in den Daten bestimmt werden. “Shape States” werden in den Sensordaten durch die orthonormalpolynomial Methode bestimmt wobei die Polynomial Koofizienten als Formdeskriptor fuer die Vorlage “Shape States” verwendet werden.

Ein verteiltes Zustandserkennungs System wurde zur Darstellung fuer das praesentierte Framework und als Werkzeug zur Ueberpruefung der vorgeschlagenen Methoden implementiert. Einzelne Komponenten einer Bohranlage wurden dazu als Beispiel zur Anwendung der verwendet Die Zustandsbestimmung kam an einer Vielzahl von Komponenten einer Bohranlage zur Anwendung. Diese sind: das Zirkulationssytem, das Drehsystem und das Hebesystem. Die durch das System erkannten Zustaende wurden automatisch mit von Experten manuell generierte Zustaende verglichen und verifiziert. Es wird vorgeschlagen das Framework und dessen Konzepte zur Analyse der Leistung von Bohranlagen und zur Optimierung der Bohrprozesse zu verwenden.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Improving the performance of the drilling process is a big challenge in todays drilling industry. The first requirement to improve drilling performance is to measure it. Performance measurement means determining the quantitative values or weights that describe each drilling operation and the complete drilling process as resultant. For example, the duration of each drilling operation is considered a useful measure. In addition, the number of drilling operations and distributions of those operations over different well drilling phases are important measures of drilling performance.

Automatic rig state detection and recognition from sensor measurements are considered as fundamental steps for monitoring the drilling rig activities. Detecting these states gives services of drilling data analysis more aptitude to examine all actions performed by the drilling crew at the rig site. Furthermore, automatic detection provides essential mechanisms to judge the performance of the drilling machinery. Consequently, this gives the ability to perform sequence mining and analysis on particular drilling process sections.

The work presented in this thesis shows the method for detecting the drilling rig states from surface sensor measurements using a distributed framework of state detection algorithms. The system is distributed over core components where each component hosts a state detection algorithm. This gives each component a supervisory role over each sensor. The components connect to each other using a middleware. The decision on the rig activity or state is taken when information from all the sensors is received at a central decision unit. The concepts of multisensor data fusion are used to embody the suggested algorithms and concepts as a real system. The suggested framework is tested

on an oil well drilling rig system where different sensor data is available. Three main sub-systems of drilling rig are tested with the suggested framework: Rotary System, Circulation System, and Hoisting System.

1.2 Problem Statement

Rig state or activity is any operation performed by drilling crew at rig surface using rig equipment in order to drill a well in the ground. The terms "rig state" and "rig activity" are interchangeably used in this thesis. The usual drilling rig states and activities are: Drilling operation (rotary and sliding); Hole cleaning operation (reaming up/down, Circulating); Making a connection to attach a new drill-stand to the drill-string; Making disconnection to remove a drill-stand from the drill-string; pulling/lowering the drill-string (out/in) the hole and putting it (in/out) the slips. The sensor measurements; which are taken from the rig and available through the mud-logging systems, are: Load on hook, torque and revolutions of the rotating drill-string, flow and pressure of the mud pumps, travelling block position, weight on bit, rate of formation penetration, hole depth and bit depth.

The main research question of this thesis is: Is it possible to detect the states and activities of a drilling rig from surface sensor measurements? If yes, how should the detection process be performed? How can the start and end timestamps of each rig state be specified? How accurate are those timestamps?

How the sensor data can be acquired and transferred from the rig site (offshore/land) to the processing center which hosts the state detection process? What kind of distributed systems is required to do the data transfer and processing operations in a reliable manner?

For each detected rig state or activity, what are the required sensors to detect it? What is the required information (features) extracted from sensors to detect all rig states? If sensor information is not sufficient then what kind of information is required to detect the rig states successfully? What is the minimum frequency of the data sampling that should be applied on each sensor data to detect rig states? What is the uncertainty of each detected state? Is it possible to evaluate the overall uncertainty of the detection process of all rig states?

Under which conditions should the detection process work? Must all the data from the beginning of drilling the well be available before starting the detection process? Or it is possible for the detection process to run at any given time during well drilling activities?

What are the required parameters for the detection process? Are those parameters time-dependent, rig-dependent, or mixture of time and rig-dependent?

1.3 Outline of the Thesis

This thesis is structured in three main parts. The first part gives basic information and literature survey about drilling rig system, systems for rig activities detection, distributed multi-sensor fusion systems, and time series analysis. The second part of this thesis discusses the research methodology which suggests a hypothesis to solve the problem statement. Also; this part shows the research steps in implementing the suggested hypothesis for detecting rig states and activities through a distributed multi-sensor fusion system. The last part of this thesis discusses the results of testing the suggested hypothesis on test data sets and what may result in future possible work.

Part I: Introduction and Literature Survey
Chapter 2 discusses drilling rig system architecture and previous work on rig activity detection systems. Chapter 3 focuses on distributed multi-sensor fusion systems and their architecture and how the communications are performed between the sensing nodes and fusing center. Middleware and Publish/Subscribe models are presented in this chapter as tools that can be used by the fusion system to communicate sensors states to fusion centers. Time series analysis techniques are presented in Chapter 4, where the terms of time series clustering and segmentation are discussed in depth.

Part II: Rig State Detection
Chapter 5 mainly demonstrates the suggested hypothesis to solve the problem statement of this thesis through dividing the problem into smaller pieces on the levels of rig sub-systems and then considering the rig state through fusing the states of all sub-systems. Chapter 6 presents detection of rig state through statistical clustering of sensor data, and how a thresholding concept is used on sensor data to detect the state of rig through detecting different states of hoisting system, circulation system and rotary system. Chapter 7 shows how a Piecewise Linear Approximation algorithm is used to detect trend states in sensor data, and shows how the two steps of filtering and trend tracking may be performed together in one step. Chapter 8 discusses how to detect states as shapes on sensor data through extract features based on moments invariants. These shapes will be used to validate specific states of rig (InSlips/MakeConnection). Chapter 9 demonstrates a distributed multi-sensor architecture to fuse all the information collected and extracted from sensor data to enable informed decision to be made about the actual state of the rig.

Part III: Conclusion and Future Work
Chapter 10 mainly summarizes the results of applying the suggested hypothesis on testing data sets and validating all the questions suggested in the problem statement. Advantages and disadvantages of the work suggested are presented. Also; the possible future work is discussed in this chapter.

Appendix A presents how data is acquired from a rig site through the distributed architecture called WITSML bridge. The WITSML standard is the main technology used in the oil industry to transfer and exchange data between different parties and is discussed in detail in this chapter.

Appendix B contains the author's published papers in international conferences and journals.

1.4 Original Work

The main contribution of the work presented in this thesis is related to the domain of distributed multisensor data fusion and its application.

The following points summarize the contribution:

- Applying clustering and segmentation algorithms to detect all possible states (activities) in sensor data of a drilling rig. An approach to detect InSlips and OutOfSlips general states in hookload sensor data using Expectation – Maximization and Envelope Algorithms is applied. Then PLA algorithm detects all the states of hoisting system on block position sensor data. Afterwords the smaller states of rotary and flow subsystems are detected on flow and rpm sensor data. The general output of this approach shows how to monitor and detect all rig states from sensor data.

- Application of the distributed fusion systems using middleware as a communication infrastructure between sensors and fusion center.

- Tracking specific borders of sensor data clusters using Expectation Maximization Algorithm. For some special cases in sensor data, an envelope algorithm is suggested to track cluster state borders more accurately than EM algorithm. Numerical verification is performed on artificially-synthesized and real rig data.

- An improved version from Piecewise Linear Approximation algorithm is suggested where similar trend segments are merged together before stepping into the final results.

- Application of orthonormal polynomial moments is performed through describing shape states in sensor data using the coefficients of orthonormal moments.
- The suggested framework in this thesis is tested and applied on different sensor data collected from different types of real drilling rigs (Offshore and Land rigs) running in different parts of the world.

Part I

Introduction and Literature Survey

Chapter 2

Drilling Rig System

2.1 Introduction

This chapter presents an overview of oil well drilling rig systems and the real-time sensor data that can be obtained from them during the drilling process. A survey on different approaches for real-time data processing in order to interpret rig states is also reviewed in this chapter.

2.2 Drilling Rig

Oil well drilling is the process of making a hole in the ground in order to extract oil, gas or any other natural resources from the subsurface; usually performed by a rig. Drilling rigs can be large structures that house equipment used to drill water wells, oil wells, or natural gas extraction wells. Numerous sensors are mounted at the rig to record different physical measurements during drilling such as block position, hookload, mud pumps flow rates, mud pumps pressures, hole depth, bit depth and torque, amongst [1]. Figure 2.1 represents a general diagram of a drilling rigs with its subsystems.

2.2.1 Hoisting System

The hoisting system works as an elaborate pulley to lift the travelling block and remove the drill pipe. This action enables the installation of an extra length of pipe or a new drill bit. Figure 2.2 shows the hoisting system at a drilling rig. The hoisting system consists of the derrick, traveling and crown blocks, the drilling line, and the drawworks. The drilling rig uses a derrick to support the drill bit and pipe (drill string). The derrick

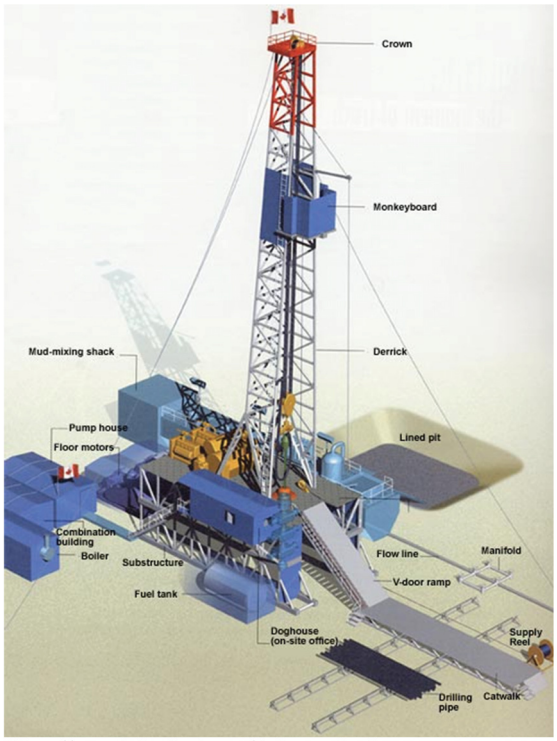


FIGURE 2.1: Drilling Rig

is a steel tower that is used to support the traveling and crown blocks and the drill string. There may be no more identifiable symbol of the oil and gas industry than the derrick on a drilling rig.

The crown and traveling blocks are a set of pulleys that raise and lower the drill string. The crown block is a stationary pulley located at the top of the derrick. The traveling block moves up and down and is used to raise and lower the drill string. These pulleys are connected to the drill string with a large diameter steel cable.

The cable is connected to a winch or drawworks. The drawworks contains a large drum around which the drilling cable is wrapped. As the drum rotates one way or the other, the drilling cable spools on or off the drum and raises or lowers the drill string.

2.2.1.1 Hookload Sensor

A Hookload sensor measures the weight that is carried by a rig's hook. The measure depends on the type of the sensor. In the case of a clamp-on sensor mounted to the deadline, the hook load readings indicate the sum of the weight of the hook itself and the weight carried by hook. If the sensor is a load pin mounted to top drive then the measurements represent the weight hanged at the hook.

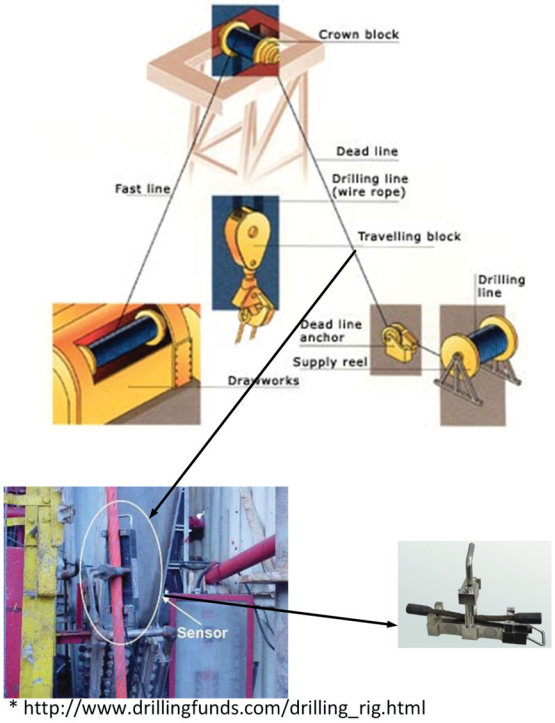


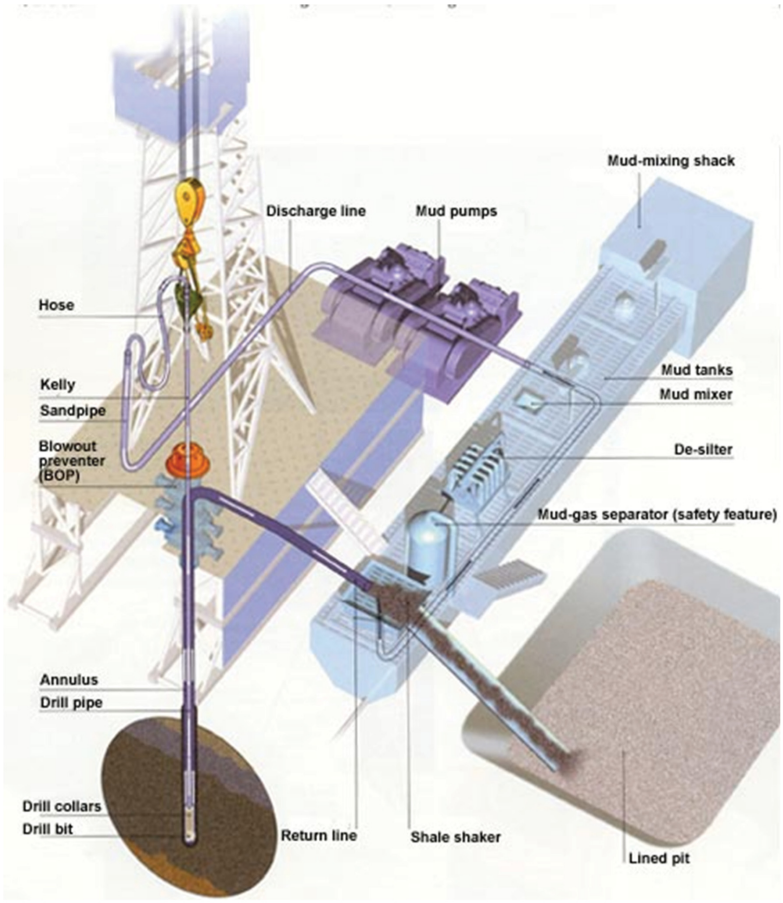
FIGURE 2.2: Hoisting System

2.2.1.2 Position of Travelling Block Sensor

The Block Position sensor measures the distance between the travelling block and the rig floor. Usually this can be measured by counting the revolutions of the drawworks multiplied by the distance of the circumference reel of the drawworks. A proximity sensor is used for counting revolutions.

2.2.2 Circulation System

A fluid called mud circulates through the drilling bit as it cuts through rock. The fluid lubricates the bit, removes rock cuttings, stabalizes the wall around the hole, and controls the pressure in the wellbore. The mud is a suspension of chemicals and minerals such as bentonite clay in water or sometimes oil. Figure 2.3 represents a systematic diagram of a circulation system at a drilling rig. Workers blend the mixture in the mud-mixing shack. The mud pumps push the fluid up the standpipe and into the drill pipe through the kelly, in the conventional rig shown here, or through fittings in a top drive mechanism. After passing through the drill bit, the mud and cuttings circulate back to the surface through the space outside the pipe, known as the annulus, and into the return line. The shale shaker, a vibrating screen, then separates the cuttings from the mud.



* http://www.drillingfunds.com/drilling_rig.html

FIGURE 2.3: Circulation System

2.2.2.1 Pumps Flowrate Sensor

Flowrate (In): The most common method of measuring flow through a positive displacement pump is to count the strokes over time and calculate the volume of each stroke. Flowrate (Out) is often measured using a flow paddle positioned in the flow line between the well and the shakers, it is common that the readings of flow out are not accurate due to the incorrect positioning of a paddle.

2.2.2.2 Pumps Pressure Sensor

The pressure readings at the standpipe and elsewhere may be measured using a diaphragm to isolate mud from a gauges hydraulic fluid. For electrical readouts, the transducer may have a diaphragm separating the mud from an electronic strain gauge package, or the mud may act directly on the transducers steel bulkhead where the bridge is attached to the bulkheads opposite side.

2.2.3 Rotary System

Conventional Drilling On most land-based rigs, a rotary table on the rig floor rotates the kelly, which turns the drill pipe and drill bit. As the drill bit penetrates deeper, the crew threads additional pipe onto the top of the drill string.

Top Drive Drilling replaces the kelly method of rotation used in conventional rotary drilling. Using hydraulic or electric motors suspended above the drill pipe enables top drives to rotate and pump continuously while drilling or during the removal of drill pipe from the hole. Most offshore units and an increasing number of land rigs use top drives.

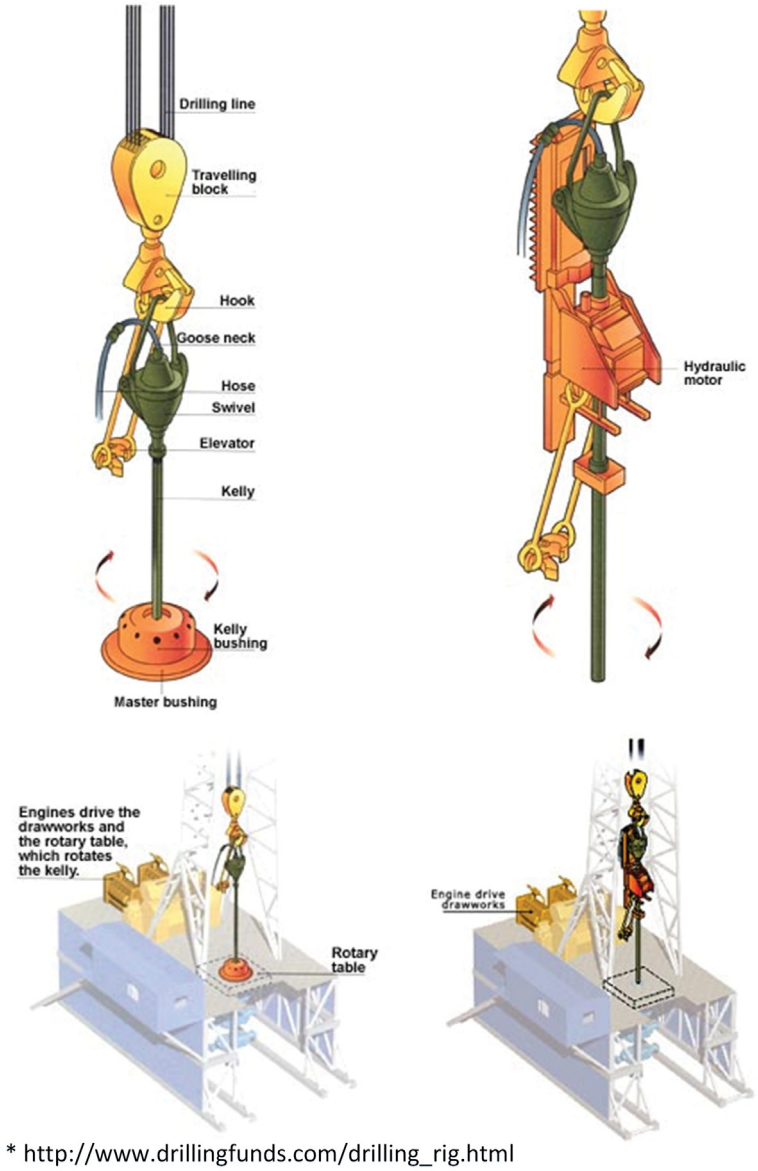


FIGURE 2.4: Rotary System

2.2.3.1 RPM Sensor

RPM measures the number or revolutions of the drill-string per minute, rpm can be measured through proximity sensors connected to rotary table or top drive for counting revolutions over time. Figure 2.4 demonstrates the two types of rotary systems that can be used at drilling rig.

2.2.3.2 Torque Sensor

Torque represents the torque force of the drill-string, this is often obtained from an electrical measurement in the powered portion of the rotary table or top drive. A common means of torque measurement on direct current (DC) rigs uses a toroidal magnetic field (a.k.a. donut) surrounding one of the power leads to the DC motor. Current passing through the magnetic field induces a voltage in the sensor. These readings are then compared to the motor manufacturers operational data [1].

2.2.4 Other Rig Readings

2.2.4.1 Rate Of Penetration Readings

Rate of Penetration represents the speed of the drill-string during a drilling operation.

2.2.4.2 Hole Depth Readings

Hole Depth is the depth of the hole drilled.

2.2.4.3 Bit Depth Readings

Bit Depth is the distance between the rig surface and the bit location in the hole, these readings help the driller to estimate the location of drill-bit inside the wellbore.

2.2.4.4 Weight on Bit Readings

Weight on Bit is calculated by subtracting the theoretical weight of the drill-string from hook load measurements, these readings helps the driller to estimate how much weight is applied on the drill bit.

2.2.5 Power System

A drilling rig needs power to operate the circulating, rotating, and hoisting systems. This power comes from two or more diesel engines. Power is transmitted to the drilling rig from either generators that provide electricity or mechanical drivers. These use a series of pulleys and belts to transmit power from the engines to the components that require the power.

2.2.6 Rig Crew

Drilling is usually done by a service company or a drilling contractor. The drilling crew is composed of a toolpusher, a driller, a derrickman, a motorman and several roughnecks and roustabouts. The toolpusher, the location supervisor for the drilling contractor, is usually a senior, experienced individual who has worked his way up through the ranks of the drilling crew positions. The driller is the supervisor of the rig crew. The driller operates the pumps, drawworks, and rotary table via the driller's console - a control room of gauges, control levers, rheostats, and other pneumatic, hydraulic and electronic instrumentation. The driller also operates the drawworks brake using a long-handled lever. Hence, the driller is sometimes referred to as the person who is "on the brake". The derrickman is in charge of the mud-processing area during periods of circulation. The derrickman also measures mud density. The motorman is responsible for engine maintenance. A roughneck is a low-ranking member of the drilling crew. The roughneck usually performs semiskilled and unskilled manual labor that requires continual hard work under difficult conditions for many hours. A roustabout is any unskilled manual laborer on the rigsite.

2.3 Drilling Rig State Detection Systems

Figure 2.5 displays sensor data acquired from a drilling rig with rig states in different colors. The data sketch shown in figure 2.5 represents the phase of tripping the drill-string into a hole and starting the drilling operations, the increases of hole depth data channel helps in recognizing the start of drilling operations. The Rig has different states over time. In this thesis, rig state can be either Reaming, Drilling, Moving Up, Moving Down, Hole Circulating or Connection/Disconnecting a new stand pipe to/from drill-string. Some people refer to these states as "Drilling Operations", in order to prevent confusion in terms between the real "Drilling" activity and the general term "Drilling Operation", it is preferred in this thesis to use "Rig States" instead of "Drilling operations".

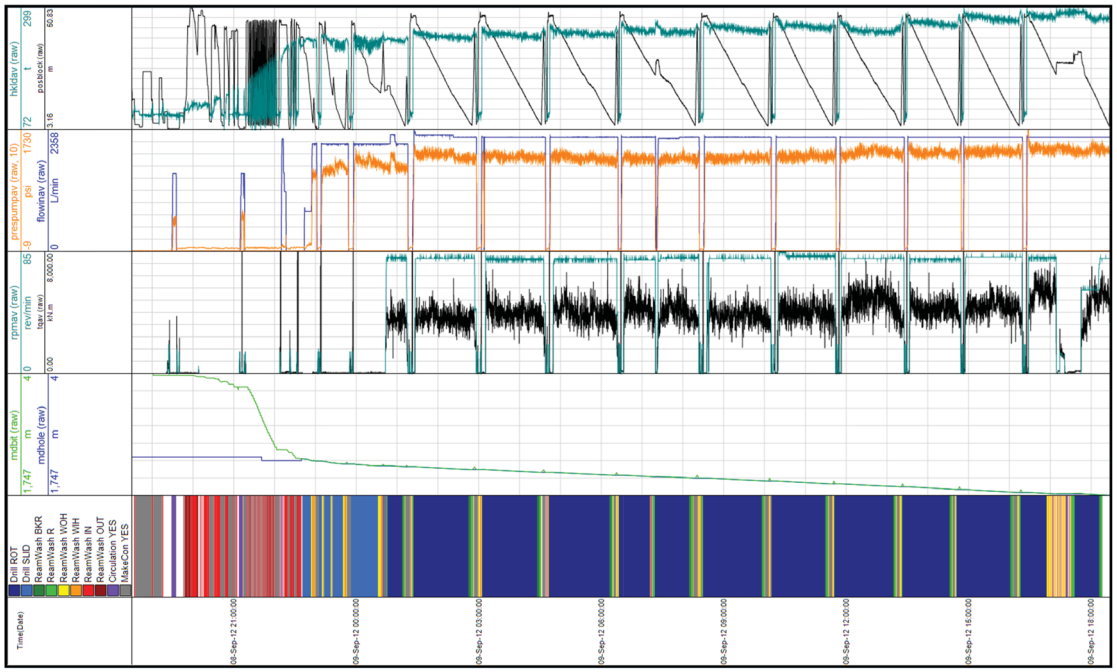


FIGURE 2.5: Rig Sensors Readings with Rig States (Drilling Operations)

A small number of groups worked on detecting drilling rig states from sensor data. Three main groups suggested systematic approaches to perform this goal. The first group used a learning approach to train an intelligent model to be able to predict rigs states based on sensors data as input [2], [3], [4] and [5]. The second group developed a state machine model that translates the sensors readings into states thus allowing the model to track those states to conclude the state of the rig [6], [7] and [8]. In order to get better results, the author suggests filtering the sensors data before processing. The third group applied the case-based reasoning concept in order to conclude rig states from sensor data [9–13].

Details on each of those approaches will be discussed in the remaining part of this chapter.

2.3.1 Learning Approach

The authors Serapiao et al. of [2],[3],[4] and [5] tried to used supervised learning techniques to train different intelligent models that can predict rig states based on sensor readings as input for these models. They used a data set of sensor data from mud-logging system. The data set is already classified manually by drilling expert and it contains the readings of Bit Depth, Weight on Hook (WOH), Stand Pipe Pressure (SPP), Drill-string Rotation (RPM) and Weight on Bit (WOB). The Data set has 3784 samples (3 days of drilling work) divided into 75% training data set and 25% testing data set. The trained

model was able to recognise the following rig states: Rotary Drilling, Rotary Reaming, Sliding Drilling, Back-reaming or Tool adjusting, Tripping, and Circulating. They used different classification algorithms on the data set such as: SVM, MLP-BP Neural Network, LWL Statistical Learning, Clonal Selection Algorithm, Parallel AIRS2 (a new version from Clonal Selection Algorithm). Figure 2.6 shows the results that reached by the learning approach on the suggested data set.

Table 4. Classification accuracy for each class in the test data

Method	Drilling operations					
	CI	TR	TA	SD	RR	RD
MLP-BP	100%	100%	96.2%	98.4%	86.0%	93.8%
SVM	100%	100%	96.6%	94.1%	83.3%	90.7%
CLONALG	0%	100%	89.5%	96.0%	72.8%	97.2%
Parallel AIRS2	100%	100%	91.7%	97.6%	75.4%	98.3%
Lazy LWL	0%	0%	91.7%	0%	93.3%	100%

FIGURE 2.6: Classification Results of Rig States - Learning Approach

The issues on their approach can be divided into two main categories (Data Set and Classification Process).

The Data Set: The used data set has 3784 samples over 3 days which means that the data is a sample each of 69 seconds. This sampling frequency is not sufficient at all to recognize some states or operations such as reaming which takes less than one min or even a few tenths of a second. The data set is considered too short and it was not clear when exactly this data was taken; was it during drilling formation phase or during tripping in or tripping out? There was no plot to show the nature of the data.
Classification Process: There is no accuracy in detection of start and end of each rigs state or drilling operation. It is not possible to distinguish between different phases of the well drilling process. The key InSlips/MakeConnection rig states or operations are not detected at all. The classification process uses sensor data directly without the extraction phase of features which makes the results unreliable for different rig types and configurations.

This approach for rig state detection using a learning approach can not be considered as a practical solution. This is because even if issues in the data set are resolved, this method is still not sufficient to detect accurate time stamps of the start and end of a rigs state. Moreover, if there is a change to concept of any rigs state, the model should be retrained for all states again. The results of suggested learning approach shows a 100%

precision which is difficult to believe especially that there are no features or preprocessing phases suggested. Determining the correct features and selecting the most important features is also considered a big challenge here.

2.3.2 State Machine Approach

The authors Mathis and Thonhauser of [6], [7] and [8] suggested a state machine approach to detect rig states from sensor data. They started with a preprocessing phase where the sensor data is filtered before any processing phase [8]. Then they used state machine models to detect different rig states from sensor data. The state machine model detects each of the following rig states: Drilling, MakeConnection, TrippingOut, TrippingIn. Each model, state machine, has the following states "YES", "NO" and "UNKOWN". The state "YES" means that the rig has the intended state i.e. the rig will be in "Drilling" state if the model of "Drilling" has "YES" state. The transitions between states ("YES", "NO", and "UNKOWN") happened through predefined thresholds on sensors readings. For examples if the hookload sensor has a reading value over predefined threshold then the MakeConnection model, state machine, switches from "YES" state to "NO" state. If the sensor data is null then the model will be switched to "UNKOWN" state. Figure 2.7 shows an example of state machine to detect "Drilling" rig state from sensor data. A collection of state machines, models, for all rig states is hosted in a rules engine. The model in the context of their work called a "Rule". Each rule has input configuration parameters to adjust its internal state to detect the required rigs state (see figure 2.7). They suggested special rules for filtering sensor data and other rules for detecting states of a rig. The rule can be a complex rule depending on other rules or a simple rule depends on a threshold. Each rule has been controlled by input parameters which are called rules configuration.

The issues on state machine approach can be summarized by the following points:

- Apply moving average filter which shifts the data, and this shifts, in turn, all detected states.
- Input sensor data should have a frequency of 1 Hz, if it is less than 1 Hz, the data will be linearly interpolated and resampled to 1 Hz data.
- Around 70 variables of configuration for each instant new thresholds should be adjusted and tested.
- Around 60 rules should be processed with around 240 States.
- Around 15 Data Channels should be configured with different buffer sizes.

- It depends on fixed hookload threshold which makes it always failed to recognize MakeConnections/InSlips.
- Drilling scenario from rig can be processed with several combinations of values for configuration parameters which makes the process of getting same recognition results impossible.
- Very sensitive to changes in block position sensor data due to heaves effects in floating rigs.

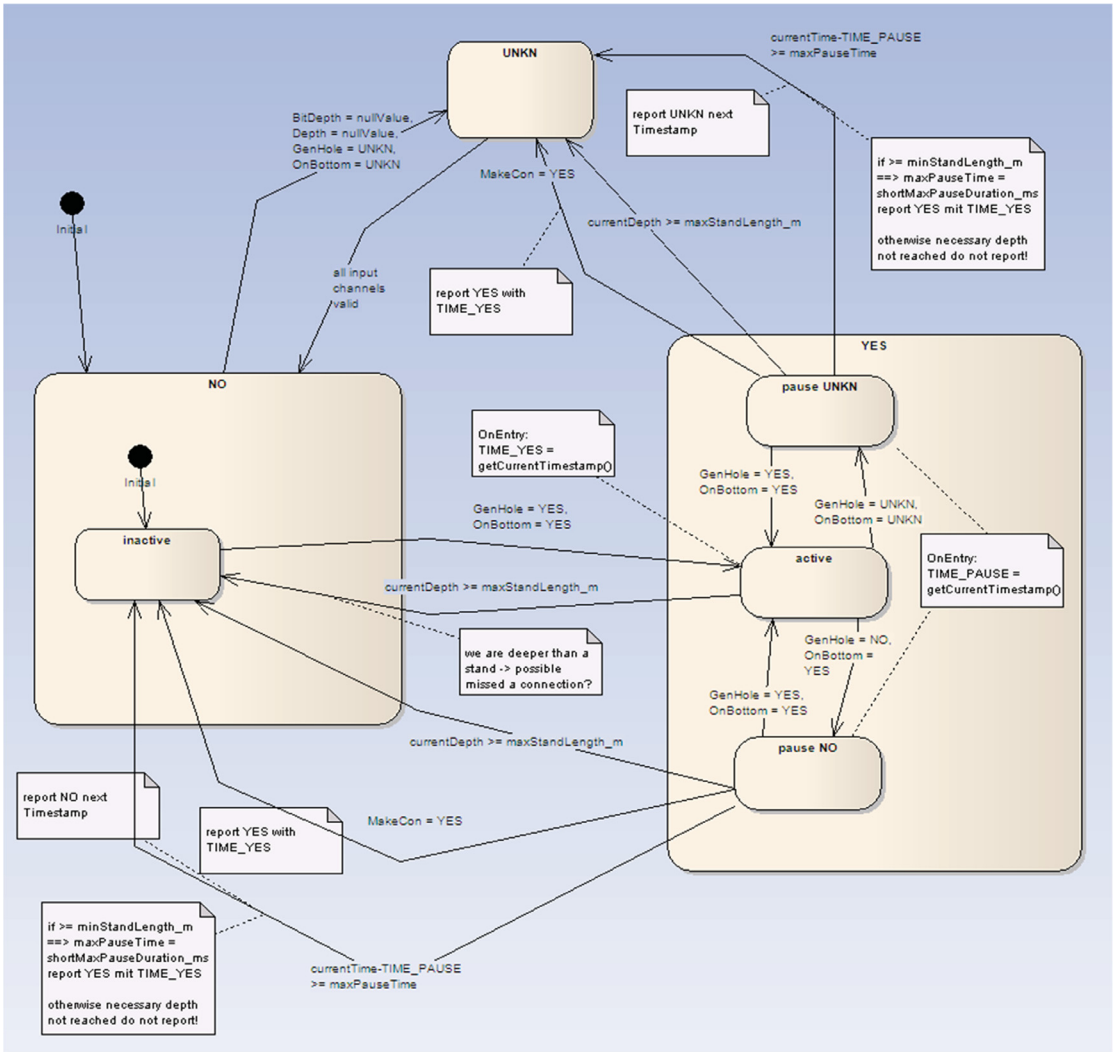


FIGURE 2.7: Drilling State Detection - State Machine Approach

Using rules engine to do a rig state detection is considered to be hard work due to the complexity of configuring the rules engine; the conguration process should be executed many times during receiving data from a rig site. This makes re-producing the same detection/recognition results from sensors data more sophisticated. Extendability of the rules engine is considered very complicated where the dependency degree between rules is very high. Figure 2.7 shows a good example of a complex rule for detecting "Drilling"

state. It is obvious that this rule has an internal state machine to detect whether the drilling bit is close to the bottom of the hole or not.

2.4 Case-based Reasoning Approach

Case-based reasoning is an approach to solving problems by reusing past experience [9]. Case-based reasoning was used as a core concept to process rig sensor measurements to help drilling engineers to understand the current drilling process situation and support them using previous similar cases [9], [10], [11], [12] and [13].

AAMODT et al. introduced a method for monitoring drilling operations [13]. This method is based on interpreting real-time sensor measurements; extracting symbolic features from these measurements; then the features are used in conjunction with a pre-defined manual input on drilling operation to formulate what they call a "case"; and then a case base should be queried to extract similar cases. Extracted cases can be reformulated and inserted into the case base as a new extension of the stored cases (storing the knowledge).

The important issues related to the work presented in [13] are:

- There is no consideration to the data quality problems such as sensor drifting, data gaps, sensor calibration, . . . , etc.
- The filtering phase is not considered in the data processing phases where the data frequency issue plays a major role in some drilling operations. For example a reaming operation can be less than 20 seconds in duration; with a data frequency of 20 seconds there is no chance to detect such an event.
- No specific details on how each data processing phase performs its functionality. For example there is a description on what the phase Activity Interpretation will do but there is no statement of how it should be implemented. The authors kept many issues open to the implementers.
- No information on what can be extracted from real-time sensor measurements.

2.5 Summary

This chapter showed a general description of the drilling rig system and its sub-systems. It gives an idea about each sensor mounted to a drilling rig and what the sensor readings mean.

Three rig state detection approaches were reviewed in this chapter, the approaches are considered as state of the art in the domains of machine learning applications and drilling operations recognition. The advantages and disadvantages of each approach were discussed and their limitations were summarized.

Chapter 3

Distributed Multi-sensor Data Fusion Systems

3.1 Introduction

The objective of this chapter is to introduce the required architecture which can be used to embody a rig state detection system. In such systems, the sensors data fusion process is required to fuse all the information acquired from sensors data and to assess the situations at the drilling rig.

In this chapter, the concepts of distributed systems and distributed computing will be presented. Then an idea of the nature of the sensors network at a rig site will be highlighted. Data fusion systems and multisensor fusion frameworks will be surveyed. The middleware as communication and messaging infrastructure in distributed systems will be discussed. Publish/Subscribe and Request/Response as communication models are defined.

3.2 Definitions

A distributed System is a software system that consists of many software components distributed on networked processing units and communicated through messages [14]. Coulouris suggested three main characteristics of distributed systems: concurrency of components, lack of global clock, and independent failure of components [14]. Andrews in [15] defined distributed computing as the usage of distributed system in solving computational problem where the problem is divided into a number of computational tasks

and then distributed over processing units which communicate with each other by message passing technique.

A rig Sensors Network is a collection of sensors mounted to different rig parts, these sensors collect and disseminate operational data on the drilling process [1]. These sensors are mounted to a data acquisition system using Ethernet network based on industrial data communication protocols such as Modbus, Profibus, TCP/IP, . . . etc.

Steinberg et al. defined data fusion as the process of combining information from different sources to provide a robust and complete description of an environment or process of interest[16]. Hall and Llinas characterized multisensor data fusion as an integration process of sensor data in order to perform a predefined mission [17]. An overview on multisensor fusion systems can be found at [16–25].

3.3 Multi-sensor Fusion Frameworks

The Joint Directors of Laboratories (JDL) Data Fusion Working Group, established in 1986, started with a four level data fusion process model [16]. Those four levels are: Object Refinement, Situation Refinement, Threat Refinement, Process Refinement. Steinberg et al. revised the JDL model and suggested a five level model to consider signal processing phase as a primary and first phase in JDL model [16]. Both JDL and revised JDL models were proposed for military applications. No clear idea on how the communication will be performed between sensor and fusion center. Phase of signal processing and sensor data alignment (Spatial and Temporal) is unclear and not well defined. Figure 3.1 represents the data fusion model of JDL (1992 version).

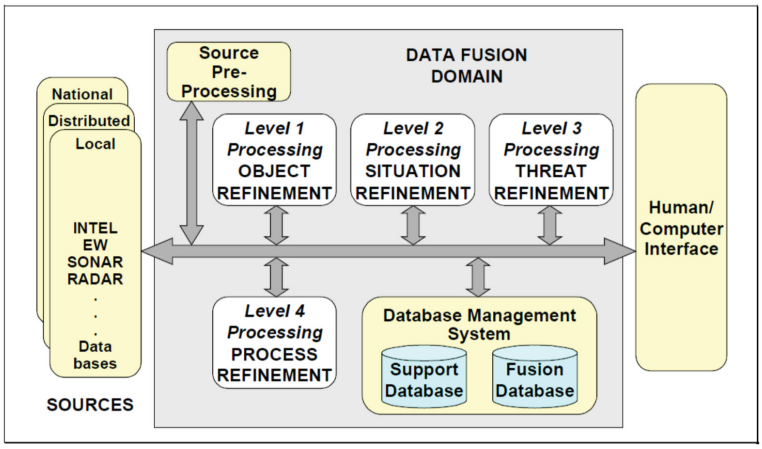


FIGURE 3.1: JDL Data Fusion Model [16]

Thomopoulos suggested a multi-sensor fusion model of three levels: Signal Level, Evidence Level, and Dynamic Level [19]. The advantage of this model over JDL model is that this model can be applied in sequential or interchangeable manner. Factors on spatial/temporal alignment of data as well as data transmission and communication channels were taken into account in this model.

Luo and Kay introduced a multi-sensor integration model through a generic data fusion structure [20]. The fusion process is performed in hierarchical manner through small fusion steps between different sensor data in context of the sensor integration process. This model suggests an interference of domain knowledge into fusion nodes through the information system.

Pau described a knowledge-based data fusion model [21]. The suggested model consists of five stages: Feature Extraction, Association Fusion, Sensor Attributes Fusion, Analysis and Aggregation, and Representation. No feedback loop between suggested levels is considered as a limitation of this model.

Harris et al. proposed a waterfall data fusion model [22]. The model demonstrates how the sensor data flows from data level to decision level. The sensor system is continuously updated with feedback information arriving from decision unit. The feedback carries control instructions to the sensor system in re-calibration, reconfiguration, and data gathering aspects. Figure 3.2 shows the idea of Waterfall Data Fusion Model.

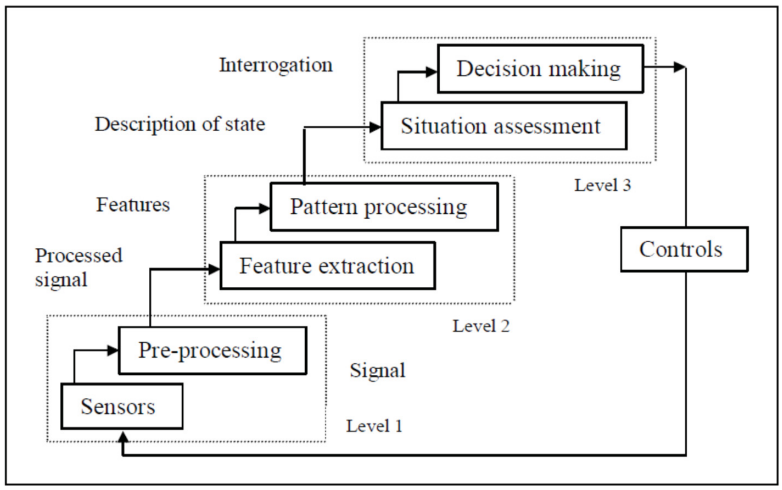


FIGURE 3.2: Waterfall Data Fusion Model as described in [22], figure taken from [26].

Schoess and Castore introduced a distributed blackboard data fusion model [24]. The model supposes a supervisor for each sensor. The sensor supervisor controls how conflicting sensor measurements are handled. This is based upon the confidence level of each sensor. Figure 3.3 demonstrates the distributed blackboard model and shows how sensor supervisors encapsulate each sensor.

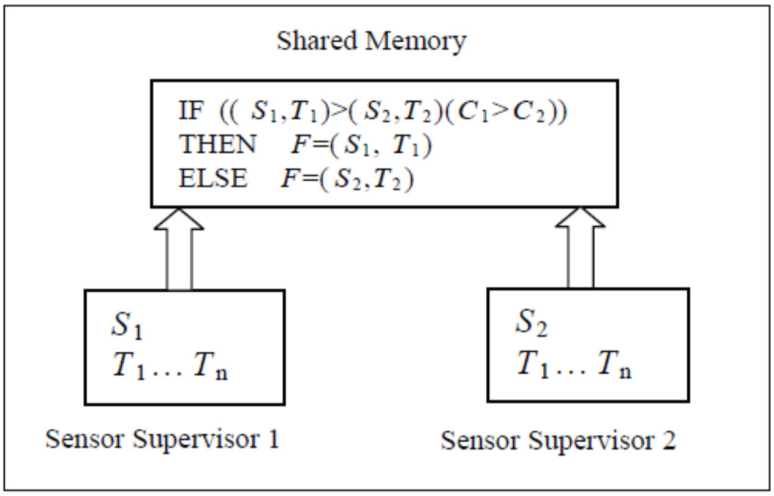


FIGURE 3.3: Distributed blackboard Model with Sensor Supervisor Technique as described in [24], figure taken from [26].

Boyd suggested a control loop for data fusion [23]. The Boyd loop was first used to model data fusion in military command processes [23]. This loop consists of four phases: Observe, Orient, Decide and Act. This model has a clear distinction from the JDL model by suggesting the phase *Act*. The *Act* phase influences the *Observe* phase with the decision taken from the *Decide* phase. This model is not clear on the concept of sensing and normalization phases in generic multi-sensor data fusion.

Dasarathy described a I/O-based fusion modes in his model for data fusion [25]. The suggested model starts from three basic levels of data fusion in most common data fusion models: Sensor Data Level, Features Level and Decisions Level. The model proposed five possible categories of transforming data between the suggested levels. These transforming categories are: [Data In - Data Out], [Data In - Features Out], [Features In - Features Out], [Features In - Decisions Out], and [Decisions In - Decisions Out] [25].

Bedworth and O'Brien describe a multi-sensor fusion model called *Omnibus* [18]. This model is a hybrid model of three other models: Boyd Loop [23], Dasarathy [25], and Waterfall [22] models. The model consists of four main modules similar to those in the Waterfall model: Sensing, Features Extraction/Pattern Recognition, Decision Making/-Context Processing, and Control Resources. The interaction between these modules is done in a closed loop manner similar to Boyd Loop, and the data processed at three main levels of Dasarathy model: Data, Features, and Decision.

3.4 Distributed System Architecture using Middleware

3.4.1 Middleware

Middleware is any software infrastructure that enables software components to communicate and exchange data in a distributed system [27]. Middleware provides interface to the software components to send or receive data in the form of messages, and this is called Message Oriented Middleware. Figure 3.4 demonstrates the concept of middleware and it shows how the interactions between applications - or components of distributed system - located at different sites can be simplified via middleware. Middleware can

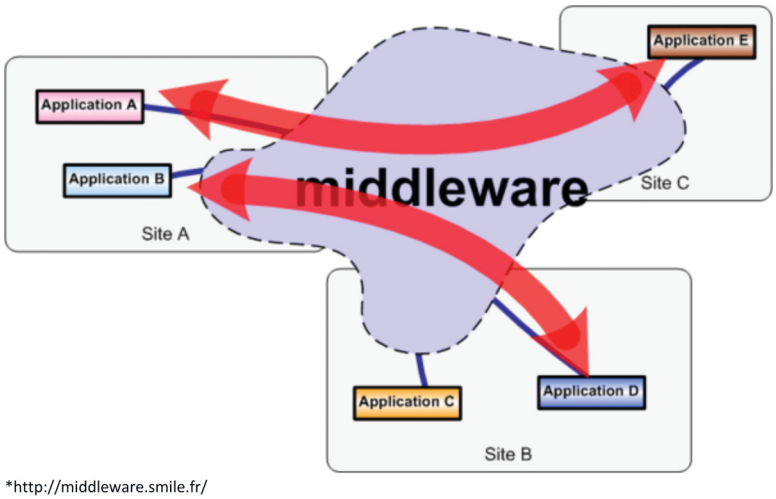


FIGURE 3.4: Concept of Middleware

be classified by the method of data exchange. The following define the categories of middleware according to [28]:

- **Message Oriented Middleware.** This is a large category and includes asynchronous store and forward application messaging capabilities as well as integration brokers that perform message transformation and routing or even business process coordination.
- **Object Middleware.** This category consists largely of Object Request Brokers.
- **RPC Middleware.** This type of middleware enables procedures on remote systems to be executed, hence the name Remote Procedure Call. Unlike message oriented middleware, RPC middleware represents synchronous interactions between systems and is commonly used within an application.
- **Database Middleware.** Database middleware allows direct access to data structures and provides interaction directly with databases. There are database gateways and a variety of connectivity options.

- **Transaction Middleware.** This category includes traditional transaction processing monitors and web application servers.
- **Portals.** It considers enterprise portal servers as middleware largely because they facilitate front end integration. They allow interaction between the users desktop and back end systems and services.

3.4.2 Request-Response Communication Model

In some references, this model is called client-server communication model [29–32]. In this communication model, the clients send requests with their data interest to a server, the server will: catch those requests; handle each request; prepare a response; and send the response back to the client. Figure 3.5 demonstrates the pattern of client-server where many clients should interact with a server to handel their requests. This architecture has a big bottleneck problem at the server node, with a large number of clients at somepoint the server will fail in processing requests [33]. To solve scalability problems many solutions were suggested; load balancing by adding more processing nodes to distribute the load has been proposed by many authors [33–36].

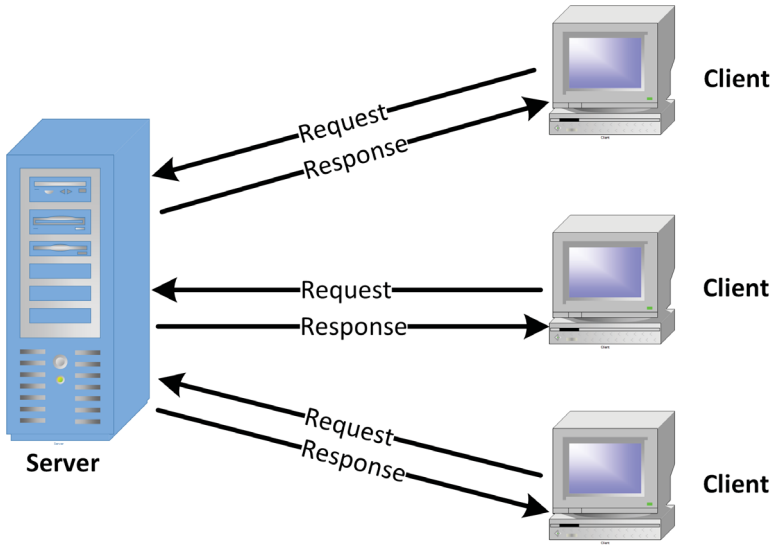


FIGURE 3.5: Request-Response Communication Model

Web services technology is considered as a state of art technology in the domain of client-server computing [37]. The W3C organization defined web services as *a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with*

other Web-related standards [38]. Web services used in a wide range of applications starting from database to web-enabled sensors. In 2008 Botts et al. suggested OGC Open Geospatial Consortium standards which suggest the concept of "Sensor Web", this concept represents a complete framework for data exchange in heterogeneous environments based on web services, the proposed framework shows how to represent each sensor as a web service and how the data flows from acquisition phase (sensor reading) to the decision making phase [39].

WITSML is a Wellsite Information Transfer Standard Markup Language is a standard for exchanging data acquired at rig site and distributed to interested parties in a standard way [40], WITSML is based on web services as the data exchange mechanism where the WITSML server exposes an interface to clients that connect to this interface and send their requests in the form of XML data queries. The server then prepares the results and sends them back to the clients. [41] (for more information see chapter A).

3.4.3 Publish-Subscribe Communication Model

Publish/Subscribe is a messaging pattern used as communication model between two software components. The component which sends the messages called publisher, and the component which receives messages called subscriber, usually the information is organized under topics, any information under these topics can be communicated to subscribers via messages, the subscribers should express their interest of getting any update under topics to publishers [42].

Figure 3.6 explains graphically how the Publish/Subscribe model is implemented between the nodes (components) where information or data is organized under topics at central network infrastructure (middleware). Then each node sends its subscriptions requests to middleware to get the information or data under specific topic. Once the topic information is updated, the middleware (publisher) sends information to the node (subscriber). Furthermore, it is possible that a node can publish information under specific topic to the middleware which, in turn, distributes the topic update in form of messages to other subscribers (nodes). Pietzuch and Bacon preferred to use the term "event" as topics update event, and they suggested the use of middleware as the event-based distribution infrastructure, where the messages will carry the events and their arguments [43]. Another important application of middleware is in real-time environments, where the events and data from distributed sensors should be communicated in real-time, it means that the messages from the publishers to subscribers should be delivered within the time constraints [44]. Heinzelman et al. describe in [45] an application of using middleware to support sensor network applications, they suggested the use of middleware called

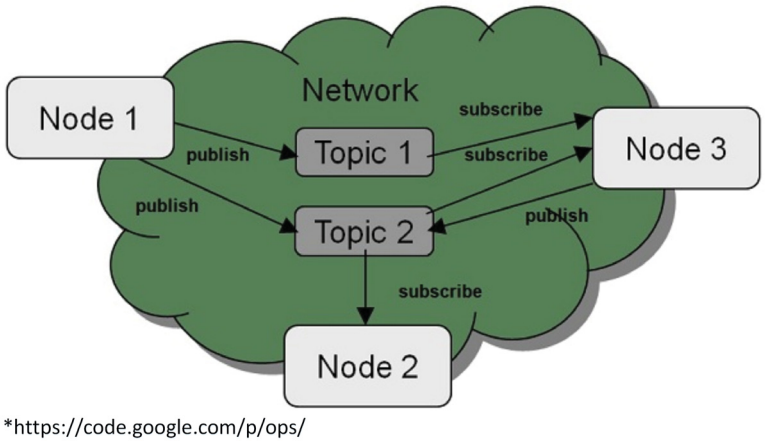


FIGURE 3.6: Publish-Subscribe Communication Model

MiLAN to fit a wide range of sensor network topologies to provide optimal events and data distribution according to a predefined Quality of Service QoS [45]. Krakowiak in the book [46] provides a comprehensive reference of applying middleware patterns in different distributed applications, the book deals with the software design patterns of object persistence, performing transactions, system management, availability, resources management and quality of service.

3.5 Summary

Client-Server Communication model using WITSML standards can be adopted to do rig sensors data communication and transfer from rig site to office site, where the communication can be performed using web services over the HTTP protocol and through a normal Internet connection.. At office site, the sensor data can be processed further using the concept of data fusion models.

In the case of a multisensor fusion system for monitoring and detecting states of drilling rigs, a hybrid multisensor fusion model of waterfall [22] and distributed blackboard [24] models can be adopted. Each sensor is associated with a supervisory component to monitor and detect sensor state. The sensor supervisor then communicates the sensor state with the fusion center as a blackboard model. The rig state detection process is located at the level of Features in Waterfall model [22]. After transferring these states to fusion center, a decision on the state of the drilling rig can be taken (see Decision Phase in the Waterfall Model [22]).

Middleware can be used as a communication infrastructure between all the components of data fusion model, where the raw data can be read from rig site and published to the middleware which, in turn, distributes the data to other components. Then all interested

components doing state detection can subscribe to the data and then publish features as results that can be used by other state detection components. The decision on rig state can be taken by state detection components and the rig states as results will be published to the middleware to be used also by other reporting or analysis components.

In this chapter, a complete review of data distribution techniques and applications is presented with the focus on the standards used on the rig site. The review is started with some definitions of basic concepts and then the idea of data fusion systems is explained. Middleware in this chapter is presented as communication and data distribution infrastructure. Two models of communications and data exchanges are reviewed in this chapter, Client-Server and Publish-Subscribe Models. At the end of this chapter, an idea of applying all the suggested concepts is proposed in order to detect rig states from sensor data.

Chapter 4

Time Series Analysis

4.1 Introduction

Time series is a collection of observations made sequentially over time [47]. Usually data collected from sensors can be described as time series, because the data represents measurements at regular time bases.

This chapter reviews the time series analysis techniques to perform three main tasks: search clusters in time series, segmentation of time series and classification of time series. In this thesis, a link between states of the rigs machines and the data clusters in sensor time series is shown, for example, two main data clusters are formed in the distribution of Hookload sensor data. In addition, time series segmentation is considered an important case in this thesis due to the mapping between the time series trend representation concept and states of rig machines. In sensors time series, some states of machines have a complex pattern or shape in sensor time series, this is considered a main reason to review the concept of time series classification. Figure 4.1 demonstrates an example on time series of rig sensors.

4.2 Time Series Clustering

The clustering is the process of identifying the structure in an unlabeled time series by objectively organizing data points into homogeneous groups where the within-group-object similarity is minimized and the between-group-object dissimilarity is maximized [48].

The data clustering process is applied in this thesis on unlabeled sensor data in order to detect the clusters that represent rig states. The main usage of clustering is to find

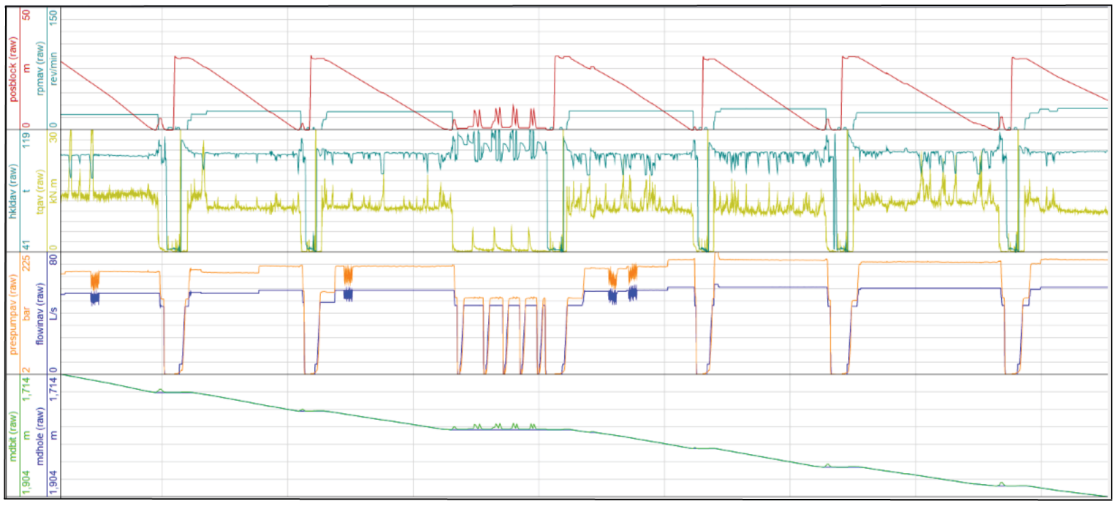


FIGURE 4.1: Rig Sensor Time Series

out the states of drill-string whether it is "InSlips" or "OutOfSlips" from unlabeled time series of hookload sensor.

The sensor time series is generated according to different states of rigs machines, each state can be viewed as separated probability distributions i.e. data points of different clusters were generated by different probability distributions [49]. If distribution family is known (Gaussian mixture, or t-distribution) then finding the clusters in a given time series is equivalent to estimating parameters of underlying models.

Propose that the prior probabilities $P(C_i)$ for cluster $C_i, i = 1, \ldots, K$ where K is number of clusters expected in the time series and the conditional probability densities $p(x|C_i, \theta_i)$ where θ_i are clusters parameters i are known. Then the mixture probability will be given as

$$p(x|\theta) = \sum_{i=1}^{K} p(x|C_i, \theta_i) P(C_i) \tag{4.1}$$

where $\theta = (\theta_1, \ldots, \theta_K)$, and $\sum_{i=1}^{K} P(C_i) = 1$. Once the parameters θ are estimated, the posterior probability for assigning a data point to a cluster can be easily calculated with Bayes's Theorem [50]. Zhuang et al. and Everitt et al. suggested use of Gaussian densities in mixture models due to their complete theory and analytical tractability [51, 52].

Duda et al. presented a Maximum Likelihood estimation as an important approach for estimation clusters parameters that maximizes the probability of generating the given time series [53]. Maximum Likelihood is given by the joint density function:

$$p(\{x_1, \ldots, x_N\}|\theta) = \prod_{j=1}^{N} p(x_j|\theta) \tag{4.2}$$

or, in logarithmic form

$$l(\theta) = \sum_{j=1}^{N} ln(p(x_j|\theta)). \tag{4.3}$$

The best estimation of parameters θ is obtained by solving log-likelihood equations $(\frac{\partial l(\theta)}{\partial \theta_i}) = 0$. Figueiredo and Jain and McLachlan and Peel showed that the solutions for the likelihood equations can be obtained under most circumstances. [54, 55]. Those circumstances are linked to the initialization process because the likelihood function of a mixture model is not unimodal (it has more than one mode), and for certain types of mixtures, the likelihood function may converge to the boundary of the parameter space (where the likelihood is unbounded) leading to meaningless estimates [54]. Furthermore, finding a solution of likelihood estimation $(\frac{\partial l(\theta)}{\partial \theta_i}) = 0$ also depends on the selection of the number of components (clusters) because this specifies the shape of the likelihood mixture function [54]. Iterative suboptimal approaches to approximate the Maximum Likelihood were suggested, one of the famous approaches for solving this is the Expectation Maximization algorithm [56].

The standard EM algorithm generates a series of parameter estimates $\{\theta^0, \theta^1, \ldots, \theta^T\}$, where T represents matching the convergence criterion, through the following steps:

1. initialize θ^0 and set $t = 0$;'
2. Expectation-Step: Compute the expectation of the complete data log-likelihood

$$Q(\theta, \theta^t) = E[log\ p(x^g, x^m|\theta)|x^g, \theta^t]; \tag{4.4}$$

3. Maximization-Step: Select a new parameter estimate that maximizes the Q-function, $\theta^{t+1} = argmax_\theta Q(\theta, \theta^t)$;
4. Increase $t = t + 1$; repeat steps 2)-3) until the convergence condition is met.

Krishnan and McLachlan and Fraley and Raftery reported that the main weaknesses of EM algorithm are: the initial parameters values, singularity of covariance matrix of the data, the possibility of convergence to a local optimum and slow performance of convergence processing [56, 57]. Krishnan and McLachlan and Celeux and Govaert showed an optimal usage of EM algorithm and K-means algorithm on data sets with a nature of Gaussian Mixture Model where K-means can be applied first to estimate clusters parameters θ_{Kmeans} and then use θ_{Kmeans} as initial parameters values in EM [56, 58].

4.3 Time Series Segmentation

Segmentation in time series is often referred as a dimensionality reduction algorithm [59]. Although the segments created could be polynomials of an arbitrary degree, the most common representation of the segments is of linear functions. Intuitively, a Piecewise Linear Representation (PLR) refers to the approximation of a time series Q, of length n, with K straight lines [60]. Figure 4.2 shows an example on Time series segmentation with its piecewise linear representation.

FIGURE 4.2: Example on Time series segmentation with its piecewise linear representation [60]

Because K is typically much smaller that n, this representation makes the storage, transmission and computation of the data more efficient. Furthermore, describing machine state based on this representation makes the decision on rig states much easier. For example, there is a need to know the movement direction of drill-string where it is Up, Down, or Static. Keogh and Kasetty described that the most of time series segmentation algorithms can be grouped into one of the following three categories [60]:

- **Sliding-Windows (SW)**: A segment is grown until it exceeds some error bound. The process repeats with the next data point not included in the newly approximated segment.
- **Top-Down (TD)**: The time series is recursively partitioned until some stopping criteria is met.
- **Bottom-Up (BU)**: Starting from the finest possible approximation, segments are merged until some stopping criteria are met.

From the applications viewpoint, Piecewise Linear Representation concept is used and applied in many applications to perform time series analysis tasks, Shatkay and Zdonik in [61] used the concept of Piecewise approximation in executing and designing time

series queries. Wu et al. used Piecewise Linear Approximation to estimate the prices trends in financial time series [62]. Amft et al. reported a successful usage of PLA on time series obtained from motion sensor in detection of eating and drinking actions [63].

4.4 Time Series Classification

Given an unlabeled time series Q, assign it to one of two or more predefined classes is a time series classification problem [64, 65].

Classification maps input data into predefined groups. It is often referred as supervised learning, as the classes are determined prior to examining the data; a set of predefined data is used in the training process and learns to recognize patterns of interest. Pattern recognition is a type of classification where an input pattern is classified into one of several classes based on its similarity to these predefined classes [64]. Two most popular methods in time series classification include the Nearest Neighbor classifier and Decision trees. Nearest Neighbor method applies the similarity measures to the object to be classified and to determine its best classification based on the existing data that has already been classified. For decision tree, a set of rules are inferred from the training data, and this set of rules is then applied to any new data to be classified [65].

The performance of classification algorithms is usually evaluated by measuring the accuracy of the classification, by determining the percentage of objects identified as the correct class [66].

4.4.1 Similarity Measures

To perform time series classification, there is a mandatory usage of distance measure [60]. Three main categories of distance measures are presented and discussed in detailed in [66]. The following list briefly describes the concepts behind each distance measures category:

Euclidean Distances One of the simplest similarity measures for time series is the Euclidean distance measure. Assume that both time sequences are of the same length n, each sequence is a point in n-dimensional Euclidean space, and define the dissimilarity between sequences C and Q and $D(C, Q) = L_p(C, Q)$, i.e. the distance between the two points measured by the Lp norm (when $p = 2$,it reduces to the familiar Euclidean distance) [66]. Figure 4.3 shows the intuition behind the Euclidean distance metric.

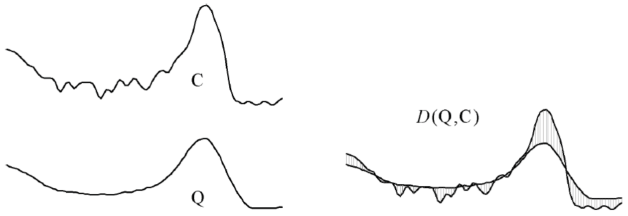


FIGURE 4.3: The intuition behind the Euclidean distance metric [66].

Dynamic Time Warping In some time series domains, a very simple distance measure such as the Euclidean distance will suffice. However, it is often the case that the two sequences have approximately the same overall component shapes, but these shapes do not line up in X-axis. Figure 4.4 shows a simple example. In order to find the similarity between such sequences or as a preprocessing step before averaging them, a warp of the time axis of one (or both) sequences is required to achieve a better alignment. Dynamic Time Warping (DTW) is a technique for effectively achieving this warping [66]. A straightforward algorithm for computing

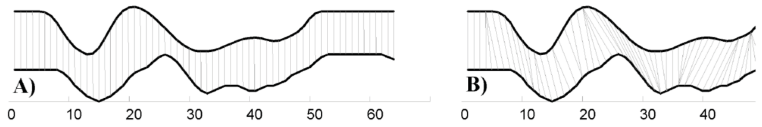


FIGURE 4.4: Two time series which require a warping measure. Euclidean distance, which assumes the i^{th} point on one sequence is aligned with i^{th} point on the other (A), will produce a pessimistic dissimilarity measure. A nonlinear alignment (B) allows a more sophisticated distance measure to be calculated [66].

the Dynamic Time Warping distance between two sequences uses a bottom-up dynamic programming approach, where the smaller sub-problems $D(i, j)$ are first determined, and then used to solve the larger sub-problems, until $D(m, n)$ is finally achieved, as illustrated in figure 4.5.

Longest Common Subsequence Similarity The longest common subsequence similarity measure (LCSS) is a variation of edit distance used in speech recognition and text pattern matching. The basic idea is to match two sequences by allowing some elements to be unmatched. The advantage of the LCSS method is that some elements may be unmatched or left out (e.g. outliers), where as in Euclidean and DTW, all elements from both sequences must be used, even the outliers[67].

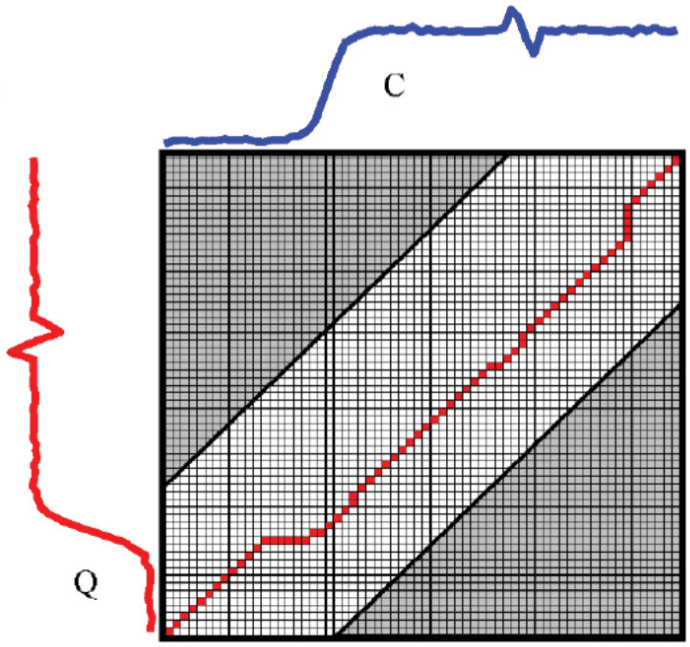


FIGURE 4.5: Two similar sequences Q and C, to align the sequences, a warping matrix is constructed, and search for the optimal warping path is shown with solid squares [66].

4.5 Summary

This chapter reviewed the literature of time series clustering, segmentation and classification. It is found that Expectation Maximization algorithm is the best algorithm that can be used to estimate parameters of data clusters in time series, the best usage of EM with K-means is to find out the initial values of the cluster parameters. Piecewise Linear Representation PLR is the state of art in domain of time series segmentation, PLR can be used to find out the main segments in a time series, the advantage of using PLR is the possibility to determine the start, end, and trend of each segment accurately. Time series classification is reviewed with distance measures that play main role in detecting some complex patterns in rig sensor time series.

Part II

Rig State Detection

Chapter 5

Research Methodology

5.1 Introduction

This chapter sheds light on the research methodology followed in this thesis. It starts with research objectives and then it shows the applied research methods to perform this research. It contains a detailed description of the working hypothesis and how this hypothesis should be tested and validated. The research design to accomplish the research objectives is also discussed in this chapter.

5.2 Research Objectives

The purpose of this research is to provide a new approach in detection drilling rig states and activities from rig sensor data. This purpose should be accomplished with less required resources and with a minimum possible delay. The delay here is the time difference between the time of real measurement taken by rig sensor and the time this measurement is interpreted as state or activity. This includes the time of transfer of this measurement from rig site to decision center plus the time of processing and decision making.

More precisely, this research aims to give answers to the following main questions:

- Is it possible to detect the states and activities of a drilling rig from surface sensors measurements with accurate start and end timestamps of each detected state?
- How the sensor data can be acquired and transferred from rig site (offshore/land) to processing center which hosts the state detection process?

- Which kind of distributed system is required to perform sensor data transfer and processing operations in a reliable manner?
- For each detected state or activity, what are the required sensors to detect it?
- What are the required information (features) extracted from sensors to detect all rig states? If sensor information is not sufficient then what kind of extra information is required to detect rig states successfully?
- What is the minimum data sampling frequency that should be applied on each sensor data to detect rig states?
- What is the uncertainty of each detected state? Is is possible to evaluate an overall uncertainty of the detection process for all rig states?
- Under which conditions should the detection process work? Should all sensor data from the point of start drilling be available before running the detection process? Or is it possible to run it from any given time point during well drilling activities?
- What are the required parameters for the detection process? Are those parameters time-dependent, rig-dependent, or a mixture of time and rig-dependent?

5.3 Research Methods

The research methods or techniques used in this thesis to perform the research work can be put into the following groups:

1. Distributed Rig Sensor Data Transfer Methods: Due to the complexity of making the intended processing system running at rig site, it is required to deal with the issue of how to get the data from rig site to the office site to be processed using high performance computers. The sensor data is usually acquired from sensors at rig site and delivered to the office site through rig acquisition system. The distributed data acquisition methods help in getting the data from the rig site to the office site. The sensor data can be stored in a database for later processing or it can be directly processed. The methods of data transfer are: Hypertext Transfer Protocol HTTP, Web Services.
2. Data Analysis Methods: These methods are used to understand the relationships between the sensor data and different rig states. The methods used here are: Time series filtering (Chebyshev Type I low pass filter), Time series clustering techniques (K-means, Expectation Maximization, Otsu), Time series segmentation

techniques (Global Thresholding, Local Adaptive Thresholding, Piecewise Linear Approximation), Shape matching and detection using polynomial moments, and Decision Trees.

3. Distributed Multi-sensor Fusion Methods: Those methods used to fuse sensor information together to conclude the final decision on rig states. Distributed blackboard multi-sensor fusion model and Waterfall data fusion model are hired for multi-sensor fusion process. Middleware is mainly used as embodiment of the suggested method for data processing.

4. Validation Methods: Those methods applied to validate the accuracy of obtained results. Confusion Matrices are used to evaluate results of applying the suggested hypothesis on testing data sets.

5.4 Research Hypothesis

To understand the hypothesis that this research is performed through, it is required to take a look at figure 5.1. This figure shows the levels of research problem. The problem is divided over four levels: Rig Level, Rig Sub-System Level, Machine Level, and Sensor Level. The research hypothesis suggests to start solving the problem from bottom to up. If the problems at sensor level are solved then they can easily be merged up to detection states at machine level. Then the states of rig sub-systems can be concluded. At the end, the state of a rig is known from the states of its sub-systems.

5.5 Research Design

The research design is the systematic steps in which the working hypothesis can be implemented and the results of research can be obtained. The research design of this thesis is a plan that specifies the tasks to obtain sensor data from a rig site and process this data to detect rig states. The following tasks form the research plan to detect rig states from sensor data:

1. Acquiring data from rig site using WITSML web service standard and distributed Middleware client bridge. Using this distributed client, a query can be sent to WITSML server at rig site then a response with sensor data will be returned back. More information on this distributed client and WITSML Standard will be discussed in the appendix as complementary work performed to explain how the data presented in this thesis is acquired.

<table>
<tr><td>Rig Level</td><td colspan="5">Detection and Monitoring Rig States and Activities using Surface Sensor Measurements</td></tr>
<tr><td>Rig Sub-System Level</td><td colspan="3">Hoisting System State Detection</td><td>Circulation System State Detection</td><td>Rotary System State Detection</td></tr>
<tr><td rowspan="2">Machine Level</td><td>Drill String State Detection</td><td>Drill String Movements Detection</td><td>Drill String Location Detection</td><td rowspan="2">Pumps States Detection</td><td rowspan="2">Top Drive State Detection</td></tr>
<tr><td>Hookload Shape Validation (U shape)</td><td>Block Position Shape Validation (S and Z shapes)</td><td>Is Bit close to Bottom of Hole? (YES/NO)</td></tr>
<tr><td>Sensor Level</td><td>Hookload State Detection (InSlips/ OutOfSlips)</td><td>Block Position Trend Detection (Up/Down/ Static)</td><td></td><td>Flow In State Detection (ON/OFF)</td><td>RPM State Detection (ON/OFF)</td></tr>
</table>

FIGURE 5.1: Suggested hypothesis for solving problem statement using Bottom-Up approach.

2. Drill String state should be detected by applying clustering mechanism to detect InSlips/OutOfSlips states from hookload sensor data.

3. Drill String movements states can be detected through trend analysis of Block Position sensor data.

4. Detection of how the drill string is close to the bottom of the hole. This can be done through cluster analysis on the difference between hole depth and length of drill string (bit depth). The state that shows whether the drill string is close to bottom of hole will appear as a separated cluster in the depths difference.

   At this point, the state of hosting system can be determined whether the drill string is InSlips or it is connecting to rig hook and whether the drill string is moving up/down/static and whether the bit is close to the bottom of hole.

5. Detecting states of circulation system by segmenting flow in sensor data using local adaptive thresholding algorithm.

6. Detection the state of rotary system by analyzing rpm sensor data and segmented into On/Off states using local adaptive thresholding algorithm.

7. Validating of the shape of hookload sensor data when InSlips state is detected. The shape of this sensor data should has a general "U" shape. It can be that the hookload sensor data has shape of "V" as a special case from the general shape "U" where the "InSlips" state last for one or two data points. This will

be discussed in more details in Borderline cases section of “InSlips/OutOfSlips” detection process. Also block position sensor data should have the shape of “S” or “Z”. The polynomial moments can be used with the sensor data to describe the shapes and measure them against predefined templates.

8. Merging the states of all rig sub-systems to conclude the final rig state. Decision trees techniques can be used here as a mechanism to give a decision based on different detailed states.

9. Validation of obtained results with reference data sets. The concept of confusion matrices can be applied to give a clear view on the overall accuracy of the suggested hypothesis. The accuracy will be measured against already classified data sets delivered by drilling experts. Shapes accuracy will be measured against artificially generated data set after consulting drilling experts about the correctness of data templates that are used to generate the data.

Chapter 6

Rig State Detection Using Statistical Clustering Analysis

6.1 Motivation

This chapter focuses on detection of rig states from sensor data using clustering analysis algorithms. The states of InSlips/OutOfSlips, Pumps ON/OFF, Rotation YES/NO are detected from Hookload, Flowrate, and RPM sensor data.

The chapter starts with discussions of the data distribution of each sensor data. The discussions lead to the fact that the data clusters helps in recognizing the rig states. Some of rig states may spread over more than one data cluster such as "InSlips" which is mainly located at a data cluster. But due the feedback from drilling experts the real start and end of this rig state is located at another data cluster. This issue is discussed in details in chapter 8 where the concept of boundaries adjustment is used to locate the correct start and end times of "InSlips" state. Otsu, Expectation Maximization, and Local Adaptive Threshold algorithms are presented for clusters detection. The algorithms applied on one simulated data set and five real data sets streamed from different rigs (see chapter A for more information on testing data sets). The results are covered in detail in this chapter. The advantages and disadvantages of each suggested algorithms summarized at the end of this chapter.

6.2 Slips States in Hookload Sensor Data

Arnaout et al. discussed the usage of Expectation Maximization algorithm to detect InSlips/OutOfSlips states of drill-string from Hookload sensor data [68, 69]. Figure

6.1 shows hookload sensor data during drilling work of a well. The data shows two important states: the drill-string is hanging at rig floor fixed by slips, thus such a state is denoted as **InSlips**, and the drill-string is hanging at hook of rig and therefore it applies force to the hook-load sensor and this is denoted as **OutOfSlips**. Two different levels of values are formed by hookload measurements during InSlips and OutOfSlips states(see Figure 6.1). Hookload sensor data shows high level of values when the state of the drill-string is OutOfSlips, and the low level of values is shown when the drill-string is InSlips [70]. Also; when the well is drilled further in the ground, the drill-string is getting longer and heavier. This explains why the hookload measurements are getting higher with the time in Figure 6.1. The hookload sensor usually measures the weight of the drill-string together with weight of the hook; therefore the hook-load is not zero at InSlips state. Normally hookload sensor data shows the weight of hook or top drive when the drill-string is at InSlips state.

The separation of InSlips from OutOfSlips states is one of main steps of the automated drilling operations classification system [71]. Usually the drilling experts manually set a threshold value for the hookload to separate this states.

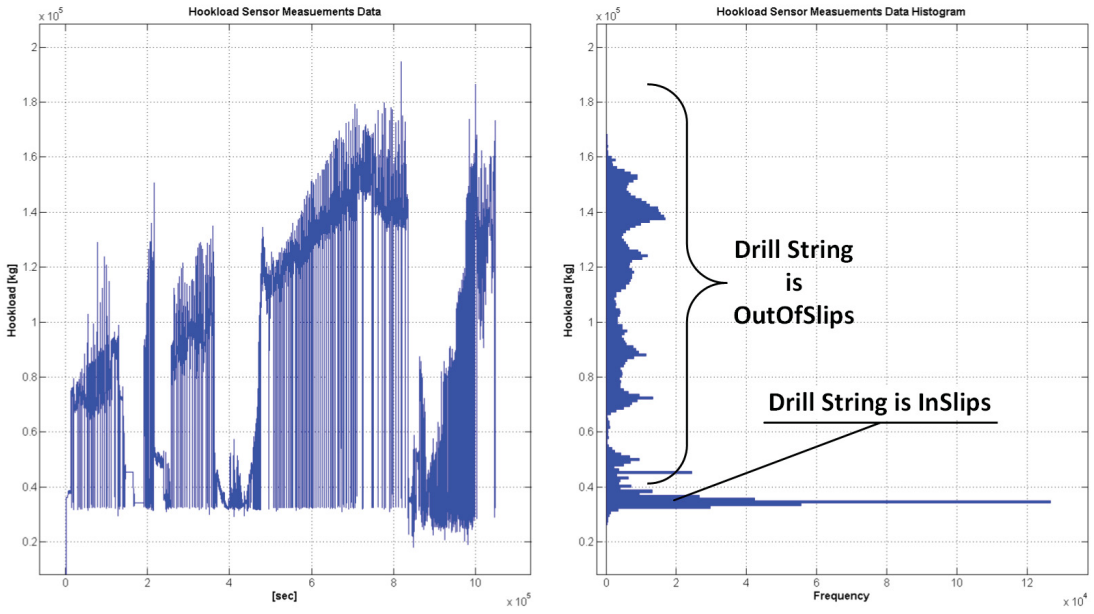


FIGURE 6.1: Hookload sensor measurements with two data clusters (Drill-string is InSlips/OutOfSlips).

The histogram in figure 6.1 shows the states InSlips/OutOfSlips. It is clear that the histogram has a nature of mixture of different clusters. The state of InSlips represents the cluster with lowest mean value. The other clusters represent the different states of drill-string when it is OutOfSlips. Those states can be drilling operations, reaming up, reaming in, moving up, moving in, circulation etc.

The main problem statement that should be solved here is that detection of InSlips states through detection of data cluster with lowest mean value. Then finding the threshold which separates this data cluster from other data clusters. When this problem is solved then both the state of drill-string and the state of hoisting system can be recognized and detected.

6.3 Pumps States in Flowrate Sensor Data

Figure 6.2 shows the flowrate measurements of the pumps that are responsible for pumping mud into hole to circulate drilling cuttings out of hole. To detect the state of circulation system, it is required to find out the states ON/OFF of those pumps. Usually the flowrate is calculated by counting the pumps strokes over time and calculate the volume of each stroke [1].

Two states ON/OFF specified on figure 6.2. Flowrate sensor data is the number of pump strokes over the last minute. This means that when the pump is turned OFF, the flowrate will reach the level zero after a period of time. The state OFF is shown as a cluster (sharp bar) at level zero on histogram but it started somewhere in the other cluster. The state ON is any value where the pump state is not OFF. The boundaries adjustment of states will be discussed in detail in chapter 8. The zero level does not mean that the values at this level should be Zero "0". Due to sensor calibration problem or sensitivity of sensors, it is possible to to have a zero level at a value different than "0". Sometimes it can be fluctuated over/below zero level with a small margin. The

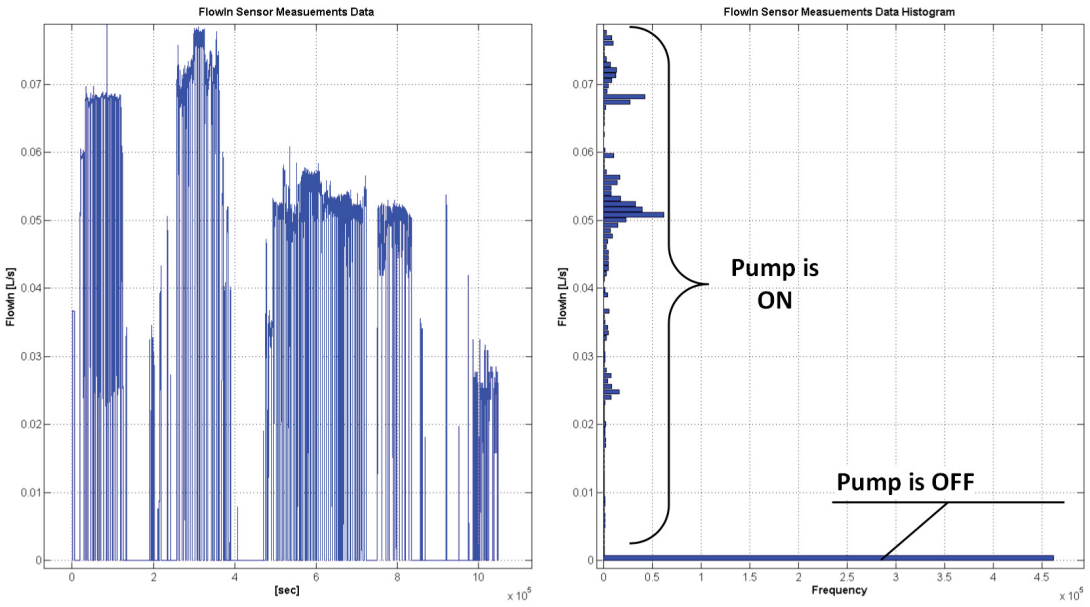


FIGURE 6.2: Flow In sensor measurements with two data clusters (Pump is ON/OFF).

data cluster with zero level of flowrate help in recognizing the OFF state of the pumps. The decision boundary of this cluster should be detected over time with consideration that the state OFF spreads to the other data cluster.

Detection of pumps state is considered as step at main level which leads to state of circulation system and then state of the rig.

6.4 Rotary States in RPM Sensor Data

Figure 6.3 shows the rpm sensor data during the well drilling process. The values can be interpreted as rotational state of drill string where the values at level zero indicates that there is no rotation or **NO** state. The values at the other levels show that the drill-string is rotating in the **YES** state.

The histogram of RPM sensor data shows two states ON/OFF as data clusters. Detection of those data cluster is an important step in the suggested approach in this thesis to recognize the rotational state of drill-string or the state of rotary system i.e. the drill-string is rotating (YES) or not rotating (NO).

The speed of a rotary table in revolutions per minute RPM are routinely measured with either an inductive proximity switch or a magnetic proximity switch or a limit switch [1]. This switch will be attached to rotary table to count the number of revolutions per minutes (RPM).

The case here is similar to the case of detecting states of circulation system. In flowrate case, the strokes of mud pump are counted per minute, but in RPM case, the revolutions of rotary table are counted per minute.

The RPM data in figure 6.3 shows values below zero level and those values came from error in RPM sensor calibration process as drilling experts commented on this case.

6.5 Data with Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities[72]. In simple words, If there is a data set, and this data set has a distribution with nature of Gaussian Mixture Model GMM, this means that there are more than data clusters - each of them has Normal distribution - mixed together in this data set.

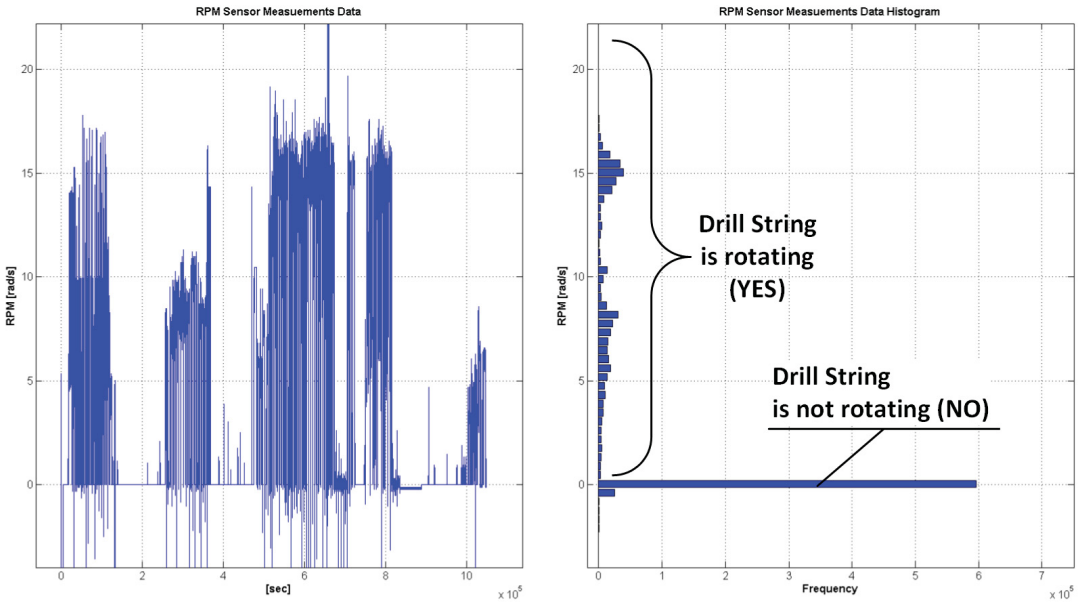


FIGURE 6.3: RPM sensor measurements with two data clusters (Drill String is rotating [YES/NO]).

Figure 6.1 shows an example on hookload data and its histogram with distribution function of Gaussian Mixture Model. Due to the prior-knowledge, the data consists of two main data clusters (InSlips and OutOfSlips). It is important to show now that each of those data clusters has Gaussian distribution with different parameters.

6.5.1 Kolmogorov - Smirnov Test

The Kolmogorov Smirnov test (KS test) is a non-parametric test for the equality of continuous, one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution (one-sample KS test), or to compare two samples (two-sample KS test). The KolmogorovSmirnov statistic quantifies a distance between the empirical distribution function of the samples and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples [73].

The Kolmogorov Smirnov test can be modified to serve as a "goodness of fit" test. In the special case of testing the normality of the distribution, the samples are standardized and compared with a standard normal distribution [73].

The test statistics equation is:

$$\max(|F(x) - G(x)|), \tag{6.1}$$

where $F(x)$ is the empirical cumulative distribution function CDF and $G(x)$ is the standard normal CDF [73].

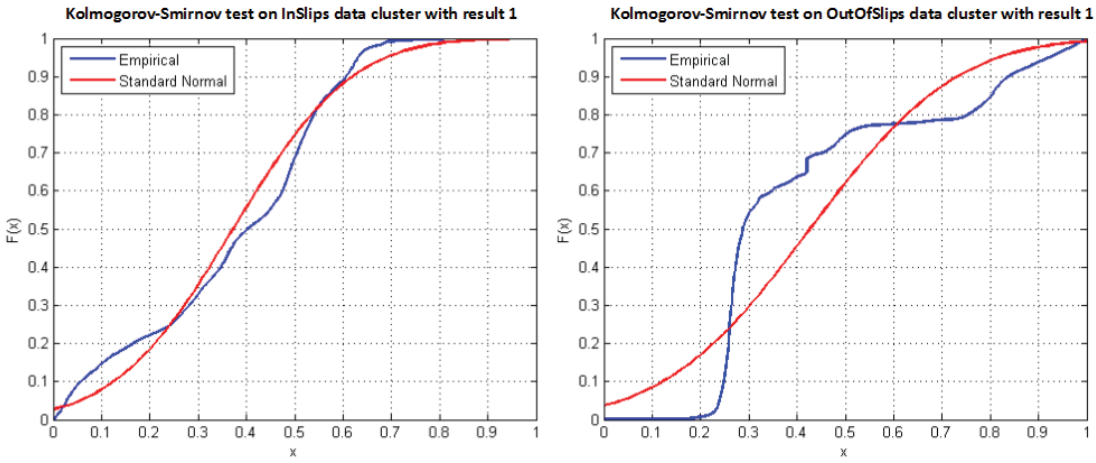


FIGURE 6.4: Kolmogorov - Smirnov Test on two data clusters of two rig states InSlips and OutOfSlips.

Figure 6.4 shows how the empirical cumulative distribution function of samples data $F(x)$ with $G(x)$ as standard normal CDF with null hypothesis. The null hypothesis is that the data samples have a standard normal distribution within a significance level 5% as suggested in [73]. The results show that each data cluster (InSlips and OutOfSlips) has a Gaussian distribution. The sensor data has the distribution with nature of a Gaussian Mixture Model.

Actually this result confirms what The Central Limit Theorem says about this issue. The Central Limit Theorem states that the arithmetic mean of a sufficiently large number of iterates of independent random variables, each with a well-defined expected value and well-defined variance, will be approximately normally distributed [74]. In the case of sensor data, each sensors data can be considered as an independent random variable because each reading (observation) obtained independently from the other readings (observations) and each of those observations has a certainty (probability) that is specified by the sensor manufacturer. Due to Central Limit Theorem, it is acceptable to say that the sensor data, which is collected over a large number of iterations, is approximately normally distributed. To match this conclusion with the rig sensor data, it can be considered that the sensor data is collected over different rig states. At each rig state (eg. Drilling state), the collected sensor data is normally distributed according to the Central Limit Theorem. And this can be extended over all other sensor data and also other rig states. This gives that all sensor data has the distribution with nature of a Gaussian Mixture Model.

6.6 States Detections using Clusters Analysis

Data clustering is the concept of grouping data samples or data points based on feature value [59]. Each data cluster refers to a specific state of system that generates this data [75]. States InSlips/OutOfSlips, Pumps On/OFF, Rotation YES/NO are data clusters in hookload, flowrate, and RPM sensor data respectively. A threshold is required to separate clusters of different states. In this paragraph, different clustering algorithms will be used to calculate different adaptive thresholds that considered as decision boundaries between the clusters i.e. separating the states.

It is shown in the previous paragraph that each data cluster in sensor data has a nature of Gaussian distribution. This means that all the sensor data sets of Hookload, Flowrate, and RPM has a nature of Gaussian Mixture Model GMM.

The shared property between three state detection problems is that always one of those states (InSlips, Pumps OFF, and Rotation NO) is one data cluster while other states (OuOfSlips, Pumps ON, and Rotation YES) have more than one data cluster.

6.6.1 Otsu Thresholding Algorithm

The Otsu algorithm is a famous thresholding algorithm in the domain of image processing. This algorithm is used to perform histogram-based image thresholding in order to reduce the gray level image to binary (white,black) image [76].

Otsu's thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the value levels each side of the threshold [76]. It finds the threshold that minimizes the weighted within-class variance. This turns out to be the same as maximizing the between-class variance.

The weighted within-class variance is:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t). \tag{6.2}$$

Where the class probabilities are estimated as [76]:

$$q_1(t) = \sum_{i=1}^{t} P(i), \qquad q_2(t) = \sum_{i=t+1}^{I} P(i). \tag{6.3}$$

And the class means are given by:

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)}, \qquad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}. \tag{6.4}$$

The individual class variances are:

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}, \tag{6.5}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}. \tag{6.6}$$

The final step in this algorithm is to run through the full range of sensor values and pick the value that minimizes $\sigma_w^2(t)$.

This algorithm can be applied on sensor data to separate states of InSlips/OutOfSlips, Pumps ON/OFF, Rotation YES/NO. This is true under the condition that there are two data clusters of data, one data cluster for each state. In the next paragraphs, this algorithm is applied on hookload, flowrate and RPM sensor data. Advantages and limitations of this algorithm will be discussed at the end of this chapter.

6.6.2 Expectation-Maximization Thresholding Algorithm

Starting form the argument which says that the nature of sensor data sets has a nature of Gaussian Mixture Model, then it will be possible to detect the states by estimating the parameters of each data cluster inside this data set. In this paragraph, Expectation Maximization algorithm is used to estimate the parameters of data clusters in hookload, flowrate, and RPM data sets. Once those clusters are known, then it will be possible to find out the intersection point of each of required cluster with its next subsequent cluster.

6.6.2.1 EM Algorithm

In case that a set of sensor measurements has nature of Gaussian Mixture Model, then Expectation-Maximization algorithm can be used to estimate parameters (mean μ and variance σ^2) of each cluster in data set. Once clusters parameters estimated, this means that system states located in sensors measurements. An algorithm is suggested (see next paragraph) to find out intersection point of clusters i.e. decision boundary of system state.

The Expectation-Maximization algorithm is an iterative optimization method for estimating some unknown parameters $\Theta\{\mu, \sigma^2\}$, given a measurements data set **D** [77]. EM mainly looks for the maximum likelihood to evaluate the parameters of statistical models [78]. Figure 6.5 shows how the Expectation-Maximization algorithm works. EM

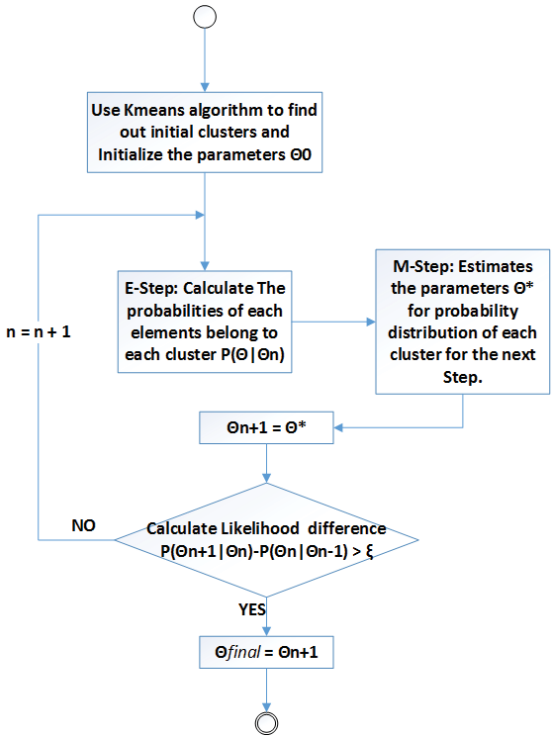


FIGURE 6.5: Expectation Maximization Algorithm.

algorithm consists of two main steps [79]:

- Expectation E-Step: This step is responsible to estimate the probability $P(\Theta)$ of each data point belonging to each cluster in the measured data D.
- Maximization M-Step: This step is responsible to estimate the parameters θ_{new} of the probability distribution of each cluster for the next step. The difference between likelihood probabilities of the new estimated parameters θ_{new} and the old parameters θ_{old} is used to measure whether the maximum likelihood probability MLP is reached or not.

Expectation-Maximization algorithm used as a core algorithm to decompose sensor data set of hookload data with distribution function of Gaussian Mixture Model into Normal-distributed data clusters. The data cluster with lowest mean value considered as InSlips data cluster. Then the intersection point with next data cluster must be calculated as a separation threshold between InSlips state and OutOfSlips state.

Clusters Intersection Point algorithm

Input:
Two univariate clusters C_1 and C_2 assumed to be Gaussian distributed with parameters set $\Theta_1 = \{\mu_1, \sigma_1\}$ and $\Theta_2 = \{\mu_2, \sigma_2\}$

Output:
The separation threshold (Intersection Point) x_i of two input clusters.

Do:
The probability density $p(x|C_k)$ for the k^th cluster of data with Gaussian distribution is given by

$$p(x|C_k) = \frac{1}{\sqrt{2\Pi\sigma_k^2}} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}. \tag{6.7}$$

According to Bayes theorem, the separation threshold x_i is located where the posterior probabilities $P(C_k|x)$ of both clusters are identical. Using

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)}, \tag{6.8}$$

$$p(C_2|x) = \frac{p(x|C_2)p(C_2)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)} \tag{6.9}$$

and the prior probabilities $p(C_1)$ and $p(C_2)$ given by

$$p(C_k) = \frac{number\ of\ points\ belonging\ to\ cluster\ C_k}{total\ number\ of\ points}. \tag{6.10}$$

The intersection point (separation threshold) x_i can be estimated by solving the equation

$$p(C_1|x_i) = p(C_2|x_i). \tag{6.11}$$

End

TABLE 6.1: Clusters Intersection Point Algorithm

6.6.2.2 Clusters Intersection Point - States Boundaries Detection

The algorithm in table 6.1 shows how to calculate the intersection point between two clusters based on Bayes theorem, using the clusters statistical parameters$\{\mu, \sigma^2\}$. The algorithm started with calculating the probability density functions of each given cluster. The threshold which separates the clusters is the point where Bayesian probability functions of each cluster are intersected i.e. the threshold is where the sum of clusters probability minimized [80].

This algorithm tracks system state in sensor measurements data. At any instance of time, it is possible to use this algorithm for cluster tracking. Furthermore, changing value of intersection point (threshold) over time considered as a dynamic adaptive threshold which helps in monitoring states of system over time.

Figure 6.6 shows the hookload data histrogram with two estimated data clusters. The intersection point is calculated and plotted as a threshold boundary between the two data clusters. It is obvious that the left clusters represents InSlips state while the right cluster represents the OutOfSlips state. It is important to mention here that this intersection points change its location due to the histogram shape and the intensity of each data cluster.

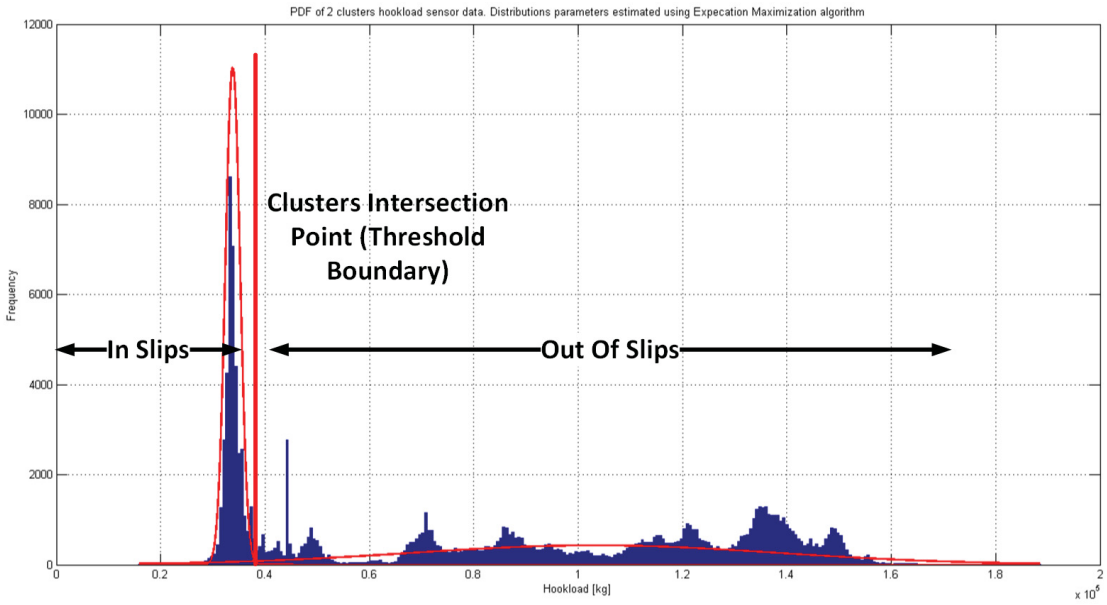


FIGURE 6.6: Estimated two data clusters on hooklosd sensor data with clusters intersection points (Threshold Boundary).

6.6.3 Local Adaptive Threshold Algorithm - Envelope Algorithm

Another algorithm suggested to track moving centers of clusters is called the Envelope algorithm. The Envelope algorithm calculates - based on a sliding window and accumulated window - upper and bottom surfaces of data, and then it tracks the middle value between the upper and lower surfaces. This value considered as an adaptive local threshold.

Table 6.2 shows the pseudo-code of the Envelope algorithm. Sliding window used to generate upper limit surface of sensor measurements data. Accumulated window applied to calculate lower limit surface of sensor measurements data. This algorithm conditioned with measurements of one sensor and two states of system. At least one of those two states is a fixed state. For each window, we use percentile of 90% on sliding window to generate upper surface values and percentile of 10% on accumulated window to calculate lower surface values. The main reason behind using percentile range of $[10\% - 90\%]$ is to make Envelope algorithm tolerant to outliers and noise in measurements data.

Envelope Calculation
Input: Sensor Data Measurements H
Output: Threshold Vector T, Upper Limit UL, Lower Limit LL
Do: Setup $Steps$ as number of iterations based on predefined window size WS. **DO LOOP** $Step = 1 : length(Steps)$ • Data Range $R = (Step - 1) * WS : Step * WS$ • $SlideWindow = H(R)$ • $AccumulatedWindow = H(1 : Step * WS)$ • $UL = prctile(SlideWindow, 90)$ • $LL = prctile(AccumulatedWindow, 10)$ • $T(R) = (UL + LL)/2$ **END LOOP**

TABLE 6.2: Envelope Algorithm

6.7 Experimental Results

To test the suggested algorithms in this chapter numerically, each of those algorithms is applied on five real data sets and on simulated data sets. The results showed good accuracy for ones and bad accuracy for others. At the end of this paragraph, a confusion error matrix summarizes the results and shows an accuracy comparison between the suggested algorithms. For more information on testing data sets, please see chapter A.

In the following paragraphs, detailed discussions will be demonstrated on each of those data sets. The thresholds calculation process uses a sliding window with a fixed start and growing end i.e. accumulated sliding window. This sensor data window is supplied to each of those algorithms and the results showed on the original data as in figures 6.7-6.12.

6.7.1 Simulated Data

Figure 6.7 shows a simulated hookload data for around 1.2 days of drilling work with 0.1 Hz as data resolution. Otsu algorithm shows a stable behavior but missed many InSlips states. Expectation Maximization missed a lot of InSlips states due to computing error of intersection point of InSlips cluster with OutOfSlips cluster. This is because the shape of histogram data is not yet formed as it supposed to be. For this reason, Expectation

Maximization algorithm failed to detect InSlips states at the beginning and end of the data set. Envelope Threshold shows a very high accuracy where it almost detects all InSlips states in the data set.

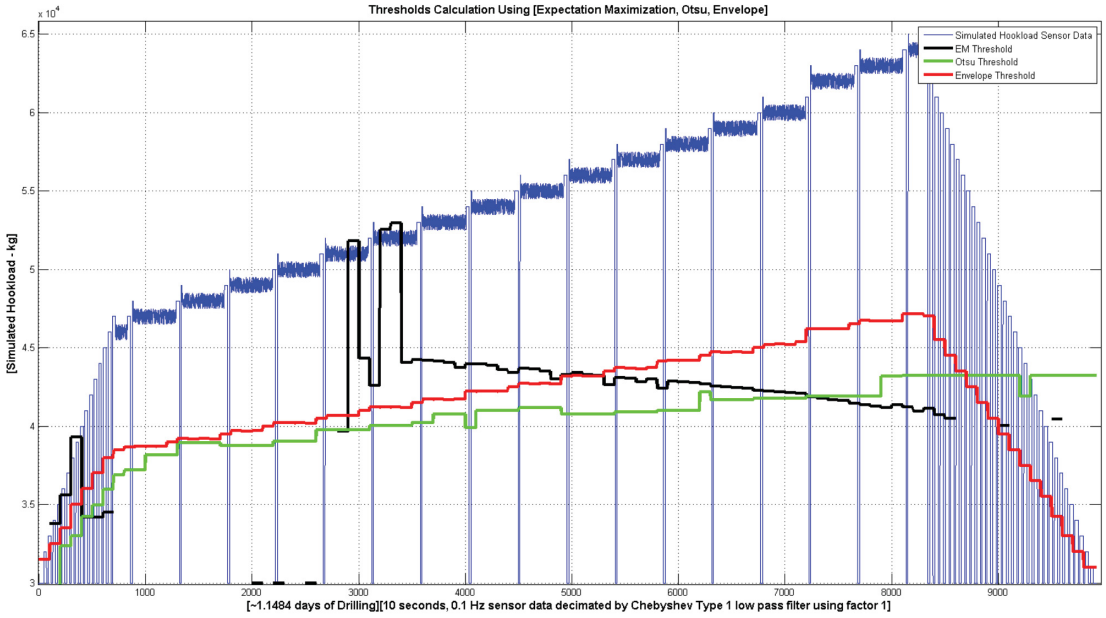


FIGURE 6.7: Application of Otsu, Expectation Maximization, and Envelope algorithms on Hookload Sensor Data - Simulated Data

6.7.2 Test Data Set 1: Rig 1, Well 1

Figure 6.8 shows application of Otsu, Expecation-Maximization, and Envelope thresholds on Test Data Set 1 (around 13 days of drilling work with 1 Hz data resolution). The results shows that Otsu threshold missed many InSlips states because it shifted up from InSlips cluster due to the data intensity of OutOfSlips cluster. Expecation Maximization algorithm failed at end part of the data set because the intersection point located very close to InSlips level and then it had false detected InSlips. Envelope Algorithm shows an interesting behaviour with adaptive values based on the upper surface and lower stable surface of hookload data. At some points, the Envelope algorithm failed in detecting InSlips because of low value of drill-string weight and this happens when the drill string pulled out of hole or when it just runned in hole. In these situations, the hookload values at InSlips state is very close to hookload values when the drill-string is OutOfSlips. And this makes the algorithms fail in detecting the correct InSlips. Another situation where the Envelope algorithm fails to detect InSlips states is when the state or drill-string is InSlips and then the drilling crew uses the hook to lift a heavy load then the sensor data shows a similar pattern like when it goes OutOfSlips.

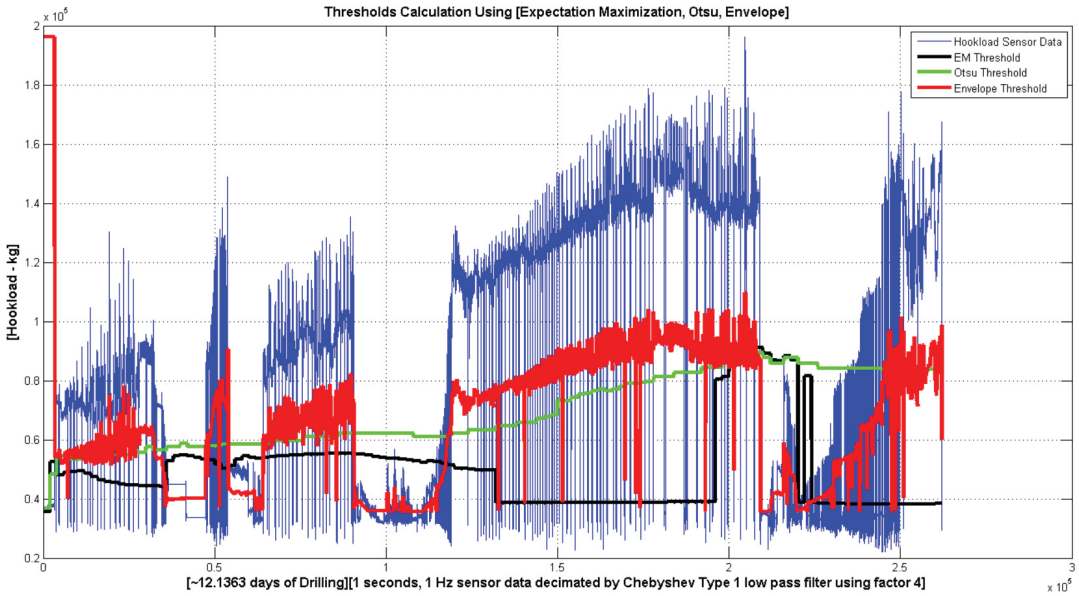


FIGURE 6.8: Application of Otsu, Expectation Maximization, and Envelope algorithms on Hookload Sensor Data - Test Data Set 1

6.7.3 Test Data Set 2: Rig 2, Well 2

Figure 6.9 shows application of Otsu, Expecation-Maximization, and Envelope threshold on Test Data Set 2 (around 26 days of drilling work with 0.2 Hz data resolution). At the beginning of the data set, it shows high values of hookload for small period of time and this is due to either stuck pipes problem or error in sensor calibration. Otsu threshold shifted up directly because of this situation in data and it takes sometime to get back to normal situation. Expectation Maximization threshold also shifted up but it was faster in getting back to normal situation. The Envelope threshold shows a very good sensetivity to such situations and get back very quick to normal situation once the high-hookload situation is over.

6.7.4 Test Data Set 3: Rig 3, Well 3

Figure 6.10 demonstrates data set 3 (around 97 days of drilling work with 0.1 Hz data resolution). The data set has cleary a hookload sensor drifting problem. This problem is obvious at the level of InSlips. Otsu threshold shows a stability over all the data but the detection of InSlips missed many states due to low values of hookload data at OutOfSlips states. This happens because the drill-string length and weight are low. Expecation Maximization threshold shows a shifting up due to data shifting up and missed number of InSlips states. Envelope algorithm shows some false detected InSlips states due to a sensor drifting problem.

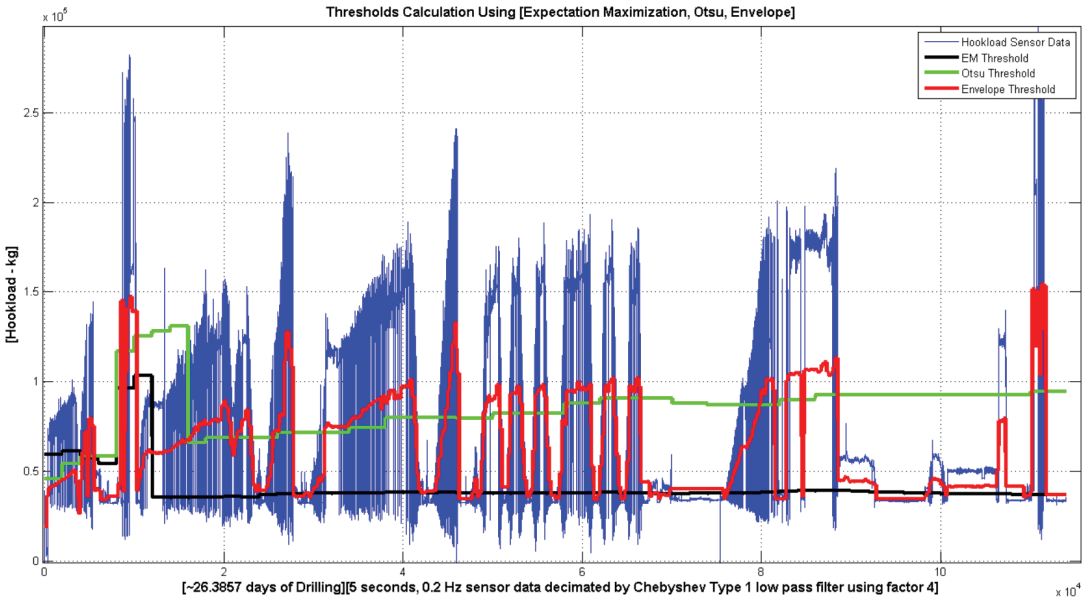


FIGURE 6.9: Application of Otsu, Expectation Maximization, and Envelope algorithms on Hookload Sensor Data - Test Data Set 2

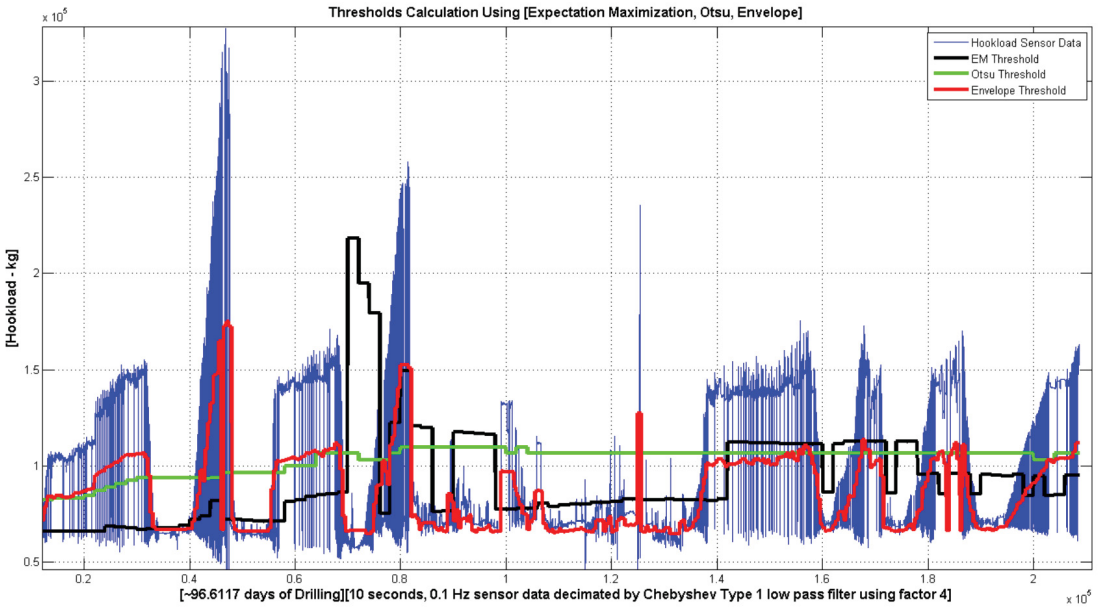


FIGURE 6.10: Application of Otsu, Expectation Maximization, and Envelope algorithms on Hookload Sensor Data - Test Data Set 3

6.7.5 Test Data Set 4: Rig 4, Well 4

Figure 6.11 shows a data set of around 40 days of drilling work with 0.1 Hz as data resolution. This data set has a problem of stuck pipe (at the end of first quarter) and hookload sensor drifting over all the period of drilling. Otsu threshold failed to detect any InSlips states for a long period of time where it is effected by the high-hookload levels due to stuck pipes problem. EM threshold shows more stable behavior than Otsu

where it got back to normal situation after a while. Envelope threshold shows a very stable behavior over stuck pipes and regarding to sensor drifting problem.

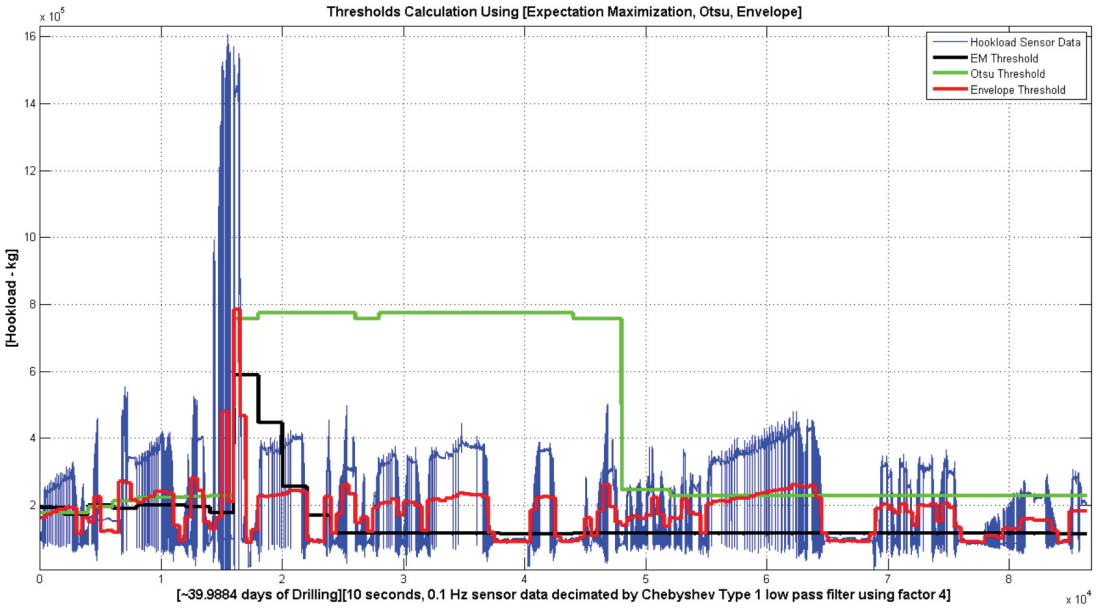


FIGURE 6.11: Application of Otsu, Expectation Maximization, and Envelope algorithms on Hookload Sensor Data - Test Data Set 4

6.7.6 Test Data Set 5: Rig 5, Well 5

Figure 6.12 shows a data set with around 25 days of drilling work and 0.1 Hz as sensor data resolution. The hookload sensor has clearly a drifting problem due to technical error. Otsu threshold missed all InSlips states when the drill-string has a low weight. Expecation Maximization threshold detects most of InSlips states but also missed InSlips when the drill-string has a low weight. But still EM is better than Otsu in this issue. Envelope algorithm detects most of InSlips and missed very few of them due to the problem in sensor drifting. Re-configuring of Envelope algorithm with new BHA weight and re-run it may cause better results. But this will make it lose more InSlips states when the drill-string has a low weight.

6.7.7 Test Data Sets: Error Matrix

The actual InSlips states for each data set is counted by drilling experts and summarized in figure 6.13. The number of correct detected InSlips for each thresholding algorithm is recorded. The error matrix technique for error rates comparison is used to compare detection accuracy of algorithms [81].

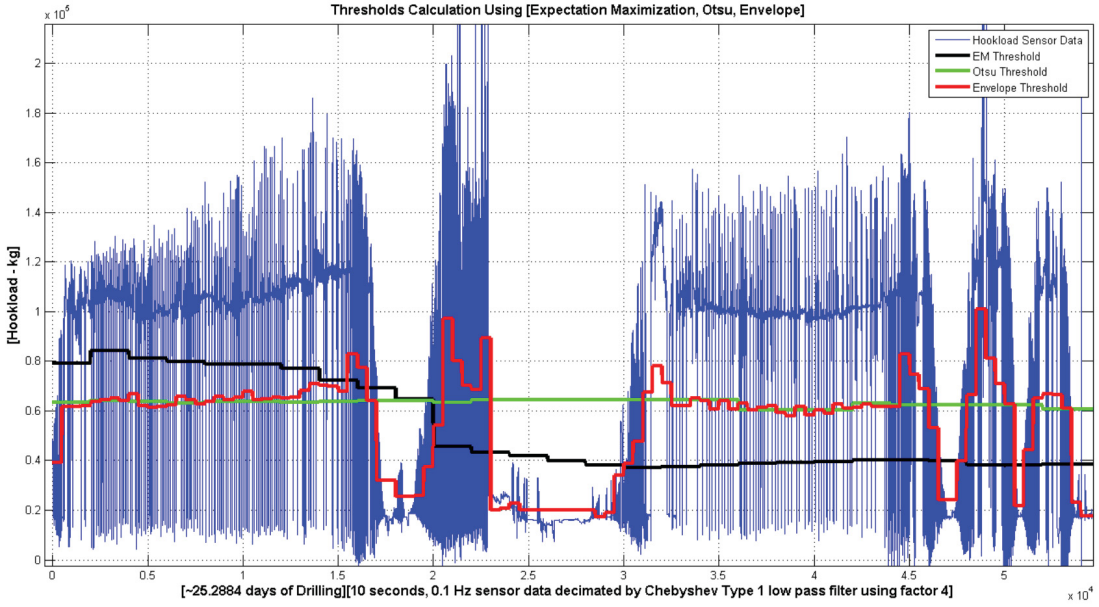


FIGURE 6.12: Application of Otsu, Expectation Maximization, and Envelope algorithms on Hookload Sensor Data - Test Data Set 5

Hookload Sensor Data Sets	Actual InSlips Count	Otsu Threshold		EM Threshold		Envelope Threshold	
		# of Correct	Error	# of Correct	Error	# of Correct	Error
Rig: 1, Well: 1	1525	435	71.48%	1145	24.92%	1143	25 %
Rig: 2, Well: 2	3819	1988	47.94%	3752	1.75%	3612	5.42%
Rig: 3, Well: 3	1685	1261	25.16%	3635	115.7 %	2420	43.62%
Rig: 4, Well: 4	1963	1455	25.88%	1716	12.58%	2060	4.94%
Rig: 5, Well: 5	3515	825	76.53%	2826	19.60%	2695	23 %
Simulated Data	66	52	21%	22	67%	65	02%
Average Error			**49%**		**35%**		**20 %**

FIGURE 6.13: Confusion Error Matrix of Applying Thresholds (Otsu, EM, Envelope) on Hookload Sensor Data from all Test Data Sets.

The results shows that Envelope algorithm has less average error rate over all test data sets. The nature of adaptive threshold calculation based on the distance between the upper surface and the lower surface is the main reason behind its high accuracy. Envelope algorithm changes the threshold based on the shape of hookload sensor data which is the key factor in differenciating the state of InSlips from the state of OutOfSlips.

Expecation Maximization algorithm has a second place in detection accuracy of InSlips state. If the shape of data histogram is not as GMM then the Expecation Maximization algorithm fails to calculate the threshold. Furthermore, EM algorithm misses many InSlips when the drill-string weight is low which means that the difference in hookload

values between InSlips and OutOfSlips states is very low comparing to the difference during drilling operations.

Otsu recorded a very high error rate and this is due to the shifting up in the threshold because of intensity of OutOfSlips cluster is more then InSlips cluster. Therefore, Otsu threshold missed most of InSlips states when the drill-string has a low weight or when there is a very high-hookload values then this threshold requires sometime to get back to the normal situation.

6.7.8 Borderline Cases

This paragraph discusses number of borderline cases to show the behaviour of the suggested thresholding algorithms with special data sets. Each borderline case is chosen to show the limitations and the advantages of the thresholding algorithm.

6.7.8.1 Case 1: Data Set with One InSlips state during Drilling Formation phase

The first data set is taken during the drilling formation phase with just one InSlips case. It started with drilling state (OutOfSlips), then InSlips state is performed before it goes to be OutOfSlips (drilling state). At the beginning of the data set, the thresholding algorithms could not recognize the OutOfSlips state. This is because the cluster of InSlips state is not formed yet. But when the InSlips data cluster started to be formed (after the first InSlips state), then the algorithms started to correctly separate the InSlips cluster from OutOfSlips cluster.

6.7.8.2 Case 2: Data Set during Tripping In phase

The data in third data set is taken at the beginning of TrippingIn phase, where the difference in hookload between InSlips and OutOfSlips states is at lowest level. **EM Algorithm:** The calculated threshold using EM algorithm is deviated to the InSlips cluster because OutOfSlips cluster is not yet formed. It started with good separation for the first InSlips, then if fails totally to do any correct recognition of InSlips state. **Envelope Algorithm:** It started with high value, which is the default value, because the difference between hookload values during InSlips and OutOfSlips is small. Then it started to recognize the correct InSlips after missing three InSlips states. **Otsu Algorithm:** The correct separation of data clusters started after the third InSlips state. The first InSlips is correctly recognized using this threshold. Otsu algorithm is less tolerance

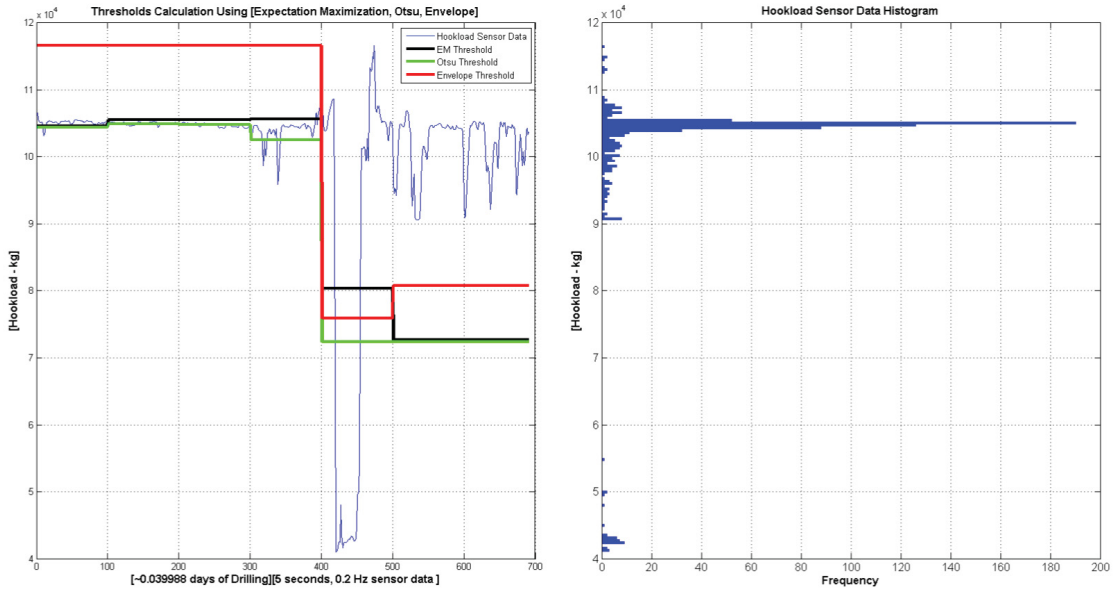


FIGURE 6.14: Borderline Case 1: Expectation Maximization, Otsu, and Envelope behaviour during data set with one InSlips state.

to the density of data clusters than EM algorithm. This is because the Otsu algorithm considers that the two clusters already exist in the data and it finds the middle value between clusters. While the EM algorithm looks to find the analytical solution of the clusters intersection point. In this case the Intersection Point located very close to the InSlips data cluster.

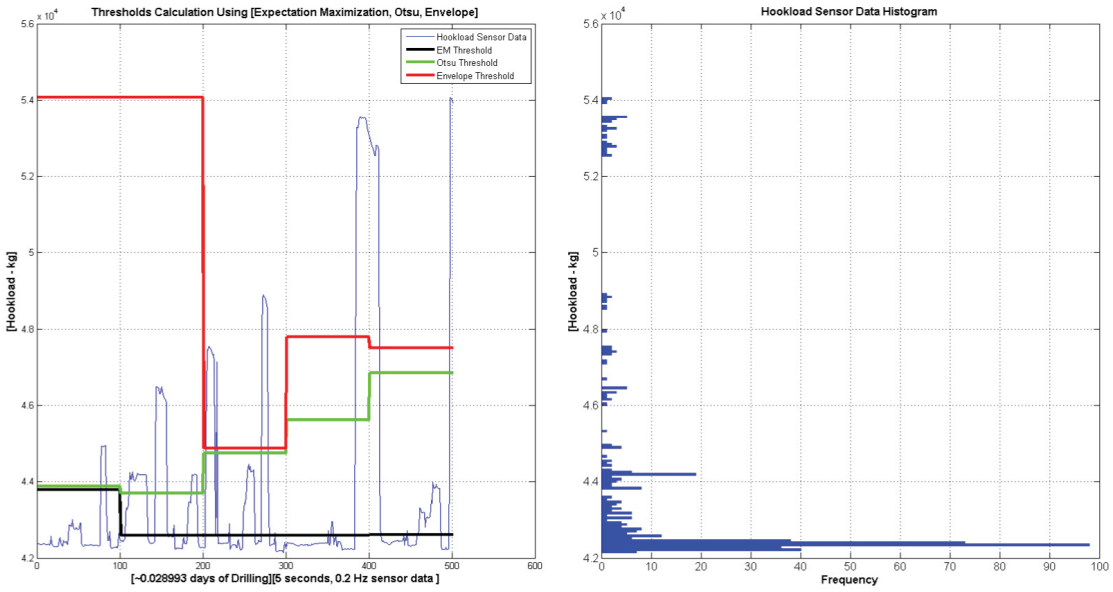


FIGURE 6.15: Borderline Case 2: Expectation Maximization, Otsu, and Envelope behaviour with data set captured during Tripping In phase.

6.7.8.3 Case 3: Data Set during Tripping Out phase

The last data set is captured during tripping the drill-string out of hole. The hookload during this phase decreases gradually as the drill-stands disconnected from the drill-string to be pulled out of hole. **All algorithms** fail to recognize the first OutOfSlips case. In addition they all show good adaptive behaviours during decreasing of hookload but **Otsu** algorithm is not affected at all by hookload values and it totally fails to recognize InSlips states at the end of Tripping Out phase. This behaviour from Otsu algorithm is because that the cluster of OutOfSlips is as dense as the cluster of InSlips. The threshold from **Envelope** algorithm shows more adaptive behaviour with the values of hookload and it recorded the most accurate recognition of InSlips state. These results of Envelope algorithm because it calculates the middle values (between the maximum and minimum limits) of hookload values over a sliding window. **EM** algorithm shows a deviated behaviour to the InSlips state because the data cluster of this state is more dense than the second one.

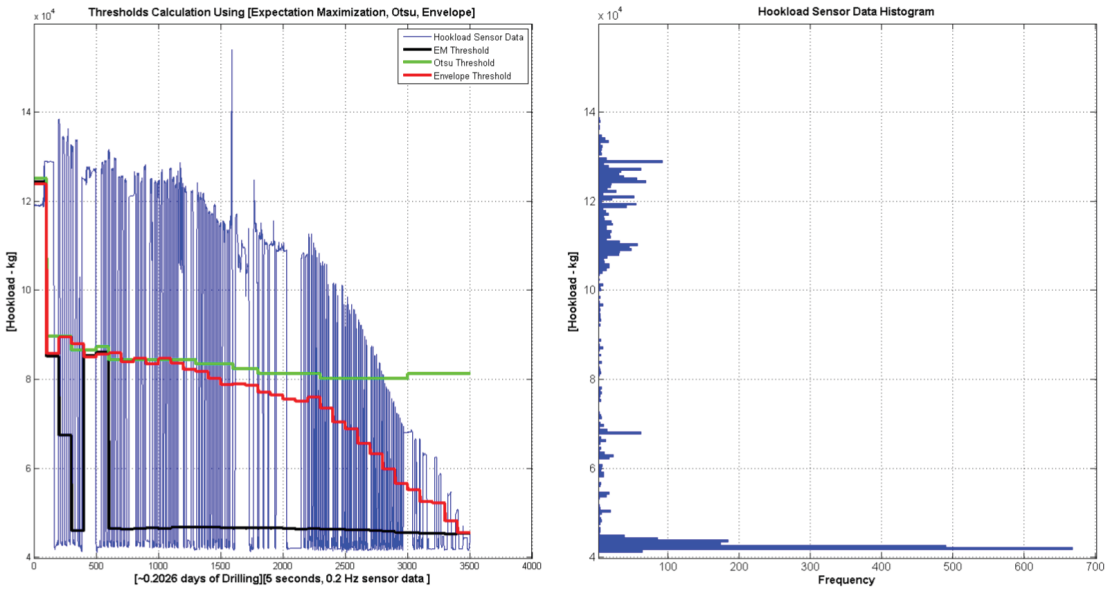


FIGURE 6.16: Borderline Case 3: Expectation Maximization, Otsu, and Envelope behaviour with data set captured during Tripping In phase.

6.8 Summary

The thresholding algorithms were studied on six data issues and they are summarized in the table 6.3. The data issues are: Stuck Pipes Problem, Sensor Drifting, Low drill-string weight, Sensitivity to Outliers, Long Data, and Short Data. Each of the suggested algorithms evaluated on scale: bad, good, and very good regarding each data issue.

Data Issue	**Otsu**	**Expectation Maximization**	**Envelope**
Stuck Pipes Problem	bad	good	very good
Sensor Drifting	good	good	good
Low drill-string weight	bad	good	very good
Sensitivity to Outliers	very good	very good	good
Long Data	good	very good	very good
Shot Data	good	bad	very good

TABLE 6.3: Summary of Hookload Thresholding Algorithms

The table 6.3 and figure 6.13 shows that Envelope algorithm is the most suitable algorithm for thresholding hookload sensor data to detect InSlips state of drill-string.

Chapter 7

Rig State Detection using Trend Analysis

7.1 Motivation

Detecting movements of drill-string is one of the major steps to detect the state of rig's hoisting system. Usually the driller lifts up or down the drill-string in order to perform one of drilling operations. Sensor data of travelling block position provides information on the location of block and how far it is from rig's surface [1].

This chapter deals with detecting the movements of drill-string from block position sensor data. It started with the description of the relationship between drill-string movement trends and state of rig. The trend can be on of the following: moving up, moving down, or static. Piecewise Linear Approximation PLA algorithm is used to detect the trends of drill-string movements. PLA is considered as state of art in time series segmentation based on trends in time series [59]. This algorithm gives start index and end index of each trend on sensor data [82]. Determination of start/end index of trends furnishes the rig state detection process with accurate information on start/end of each drilling event. The PLA is applied on block position sensor data within the boundaries of segments which come from hookload thresholding process (see chapter 7). The idea is to divide the segments that come from previous segmentation process into smaller segments. The resulted segments then will carries properties of hookload (InSlips/OutOfSlips) and block position (Up/Down/Static). This plays a major role in deciding the general state of rig.

The concept of joint confidence of estimated segments is presented in this chapter. Examples are showed at the end of this chapter on sensor data of block position. Some examples are real ones measured on real rigs and other examples are artificially-synthesized (simulated) in order to prove the concept of PLA and measure its accuracy.

7.2 States as Trends in Block Position Sensor Data

Figure 7.1 demonstrates a sketch of block position sensor data. It shows how the trends on block position are linked to drilling operations performed by drilling crew. This means that detecting trend state of drill-string gives information not only on which operation is taking place but also on what the current state of rig is.

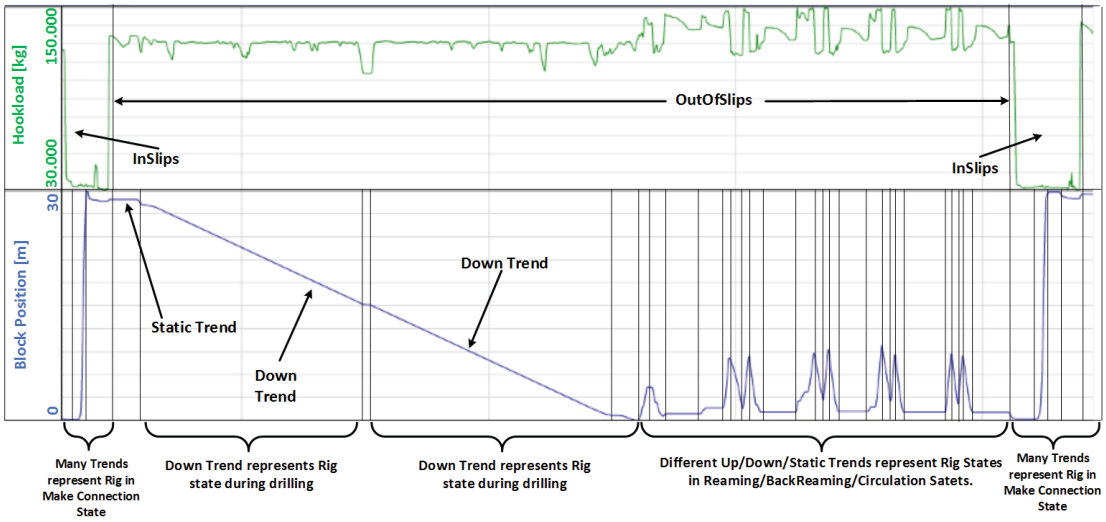


FIGURE 7.1: Relationship between trends of rig's block position sensor data and rig state.

It is obvious on figure 7.1 that each trend should be bounded by start and end borders. The trend here represents a drilling event. It is either drilling (Down), reaming up (Up), reaming down (Down), circulation (Static), running out (Up), running in (Down), wash out (Up), and wash in (Down).

7.3 Piecewise Linear Approximation

Piecewise Linear Approximation (PLA) is a tool for time series segmentation. PLA is used to approximate a time series T, of length n, with K straight lines. PLA estimates the main segments in time series [82]. If the time series represents sensor data then each of the approximated segment refers to a specific state in the measured phenomena [59].

There are three main approaches to perform Piecewise Linear Approximation concept on time series: Sliding Window, Top Down, and Bottom Up.

Sliding Window approach is worked by anchoring the left point of a potential segment at the first data point of a time series, then attempting to approximate data data to the right with increasing longer segments. The process results a new segment if an $error_{max}$ is reached, and then building a new segment will start from the end of previous segment. The whole process will continue till end of timer series. The issue in this approach is configuring the $error_{max}$ parameter [82]. If the sensor data is noisy then configuring $error_{max}$ will be a major issue in this algorithm [59].

Top Down approach of PLA considers every possible segment of the time series and splitting it at the best location. Both subsections are then tested to see if their approximation error is below some user-specified $error_{min}$ threshold. If not, the algorithm recursively continues to split the subsequences until all the segments have approximation errors below the threshold [82].

The algorithm Bottom to Up [59] forms the base for the segmentation by Piecewise Linear Approximation based on customized error cost function. The algorithm begins by creating the finest possible approximation of the time series consisting of n samples by using initially $n/2$ segments. In a subsequent step, the costs of merging pairs of adjacent segments are calculated. The algorithm iteratively merges the pairs with the lowest costs until a stopping criterion is met. Merging pairs of adjacent segments, i and $i+1$, bookkeeping about the neighborhood merging costs is inevitable. The cost of merging the actual segment with both, right and left neighbors, must be calculated [82]. Bottom Up and Top Down approaches complement each other.

The table 7.1 shows the pseudocode of PLA Bottom-Up algorithm. The important parameter $error_{max}$ regarded as the stop condition of the algorithm. The algorithm can be changed to provide the number of expected segments, then the convergence will continue till the number of estimated segments is reached. Another thing can be done to the convergence process is that a parameter of maximum iterations can be passed to the algorithm, this parameter can be very useful to limit un-expected performance issues in case that the $error_{max}$ is not reached.

Figure 7.2 explains graphically how PLA Bottom Up works. It shows the start step with creating fine $n/2$ segments where n is the length of block position data segment. Then in step 2, the lowest cost of merge is calculated and located. The cost of merge is given by:

$$merge_{cost} = \sum (DataSegment - BestFitToLine)^2. \tag{7.1}$$

Piecewise Linear Approximation - Bottom Up Approach [82]
Input: T, $error_{max}$
Output: Seg_{TS}
DO: **// Create initial final approximation** for i = 1 : 2 : length(T) Seg_{TS} = concat(Seg_{TS}, CreateSegment($T[i : i + 1]$)); end **// Find cost or merging each pair of segments** for i = 1: length(Seg_{TS}) - 1 $merge_{cost}(i)$ = CalculateError(merge($Seg_{TS}(i)$, $Seg_{TS}(i + 1)$)); end **// While not finished.** while min($merge_{cost}$) < $error_{max}$ **//find cheapest pair to merge** index = min($merge_{cost}$); **// merge them.** $Seg_{TS}(index)$ = merge(Seg_{TS}(index), Seg_{TS}(index + 1)); delete(Seg_{TS}(index + 1)); $merge_{cost}$(index) = CalculateError(merge(Seg_{TS}(index), Seg_{TS}(index + 1))); $merge_{cost}$(index - 1) = CalculateError(merge(Seg_{TS}(index - 1), Seg_{TS}(index))); end

TABLE 7.1: Piecewise Linear Approximation - Bottom Up Algorithm

BestFitToLine is the linear regression of data segment. In Step 3, the process of merging and searching for the best merge is continued till the stop condition is met. If the merging cost is greater than the maximum error then it means that stop condition is met:

$$error_{max} \leq merge_{cost}. \tag{7.2}$$

The final step in PLA Bottom Up Algorithm is to conclude start, end and trend of each segment. The trend can be either static (S), up (U), or down (D). The trend of segment S is recognized based on slope of the segment as defined in the following formula:

$$trend(S) = \begin{cases} Static, & if slope(S) = 0 \\ Up, & if slope(S) > 0 \\ Down. & if slope(S) < 0 \end{cases} \tag{7.3}$$

By applying PLA on block position sensor data, it will be possible to detect all the movements of drill-string. Those movements (start/end timestamps) and the trend of each movement (Up,Down, Static) are the main factors in deciding what is the state of the drilling rig. For example, if the drill-string is moving down, then it not possible to

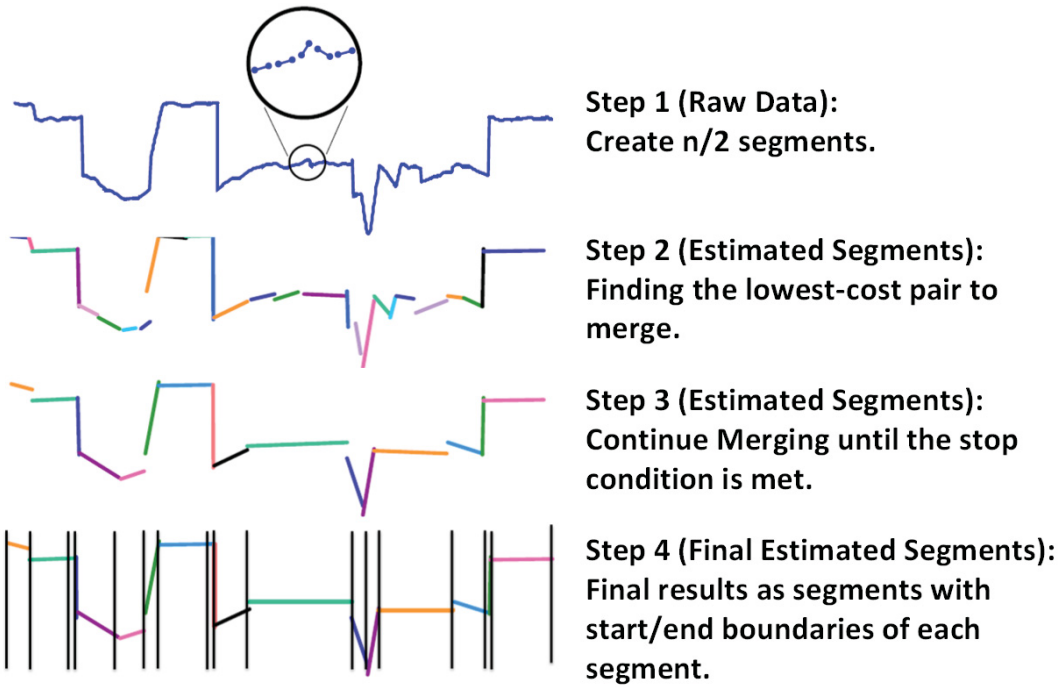


FIGURE 7.2: Piecewise Linear Approximation PLA, Bottom-Up approach.

consider this as circulation state. Another example is when the drill-string is moving up, it is then impossible to consider this state as drilling state of the rig regardless of whatever the states of circulation or rotary systems are.

7.4 Experimental Results

In this paraharph, Piecewise Linear Approximation is applied on different data sets of block position sensor data. It starts with simulated block position example, then real block position sensor data taken from real data set (Data Set: Rig 1, Test Well 1, see chapter A for more information). The last example is block position sensor data that has a noise which comes from heaves compensator effect on drilling ships.

7.4.1 Simulated Data

The first example is a time series similar to block position sensor data, it is simulated using a drilling simulator (see chapter A). The data series shows different trend states that simulate the reality on drilling rig. Figure 7.3 demonstrates the simulated block position sensor data and the results of PLA displayed as vertical lines each of which represents a border of a segment, each segment is labeled with a trend (Up, Down,

Static). It is obvious on the data set that all the possible segments on simulated block position are detected and their trend is recognized.

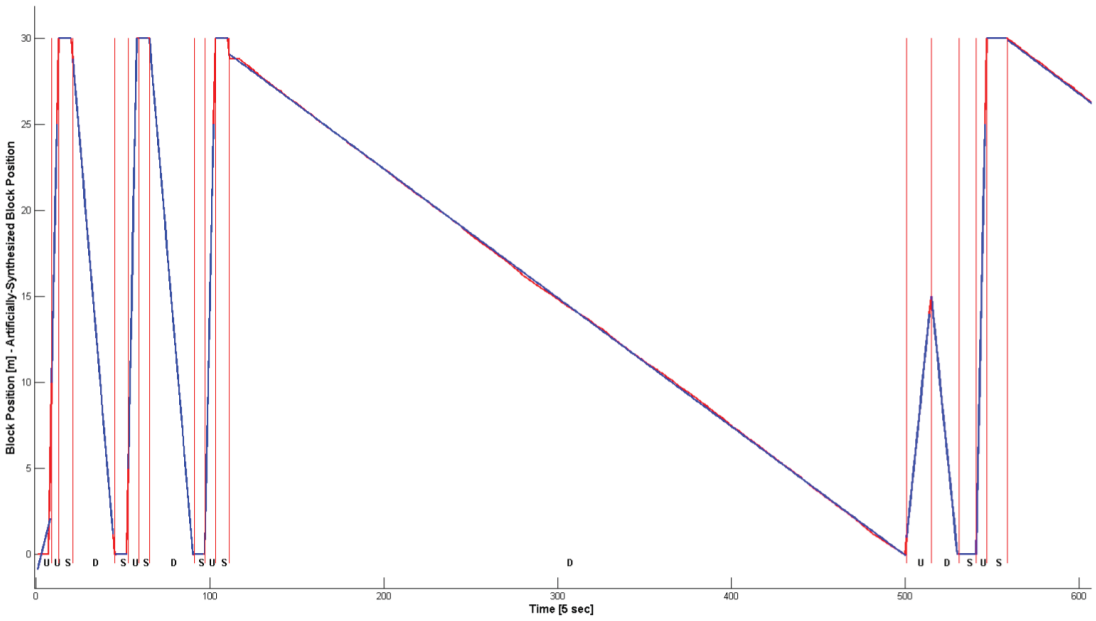


FIGURE 7.3: Piecewise Linear Approximation PLA, Bottom-Up approach applied on artificially-synthesized sensor data

7.4.2 Real Rig Data

The second example presented in this paragraph is a data section from real block position data. The testing data set from which this section is taken is Rig 1, Test Well 1 (see chapter A for more information). The results show an estimation of the main segments in block position. It is clear that many down segments (D) are consecutive, this is because that the movement of block has different speeds then all the trends are detected as down (D) segments. The same thing is true on up segments (U). Another important thing is shows on figure 7.4 is detecting static sections, those sections indicate the situations where the drilling crew does not move the drill-string but it waits for starting up the ciculation and rotary systems before beginning with drilling and moving the drill-string down.

7.4.3 Borderline Cases

Another example on block position with heave effect (noise) is presented. Figure 7.5 demonstrates block position with effect of heaves compensator on drilling ships, it is obvious that the compensation effect is on some sections of the block position, those sections are the drilling states. Figure 7.5 shows that block position has no noise when

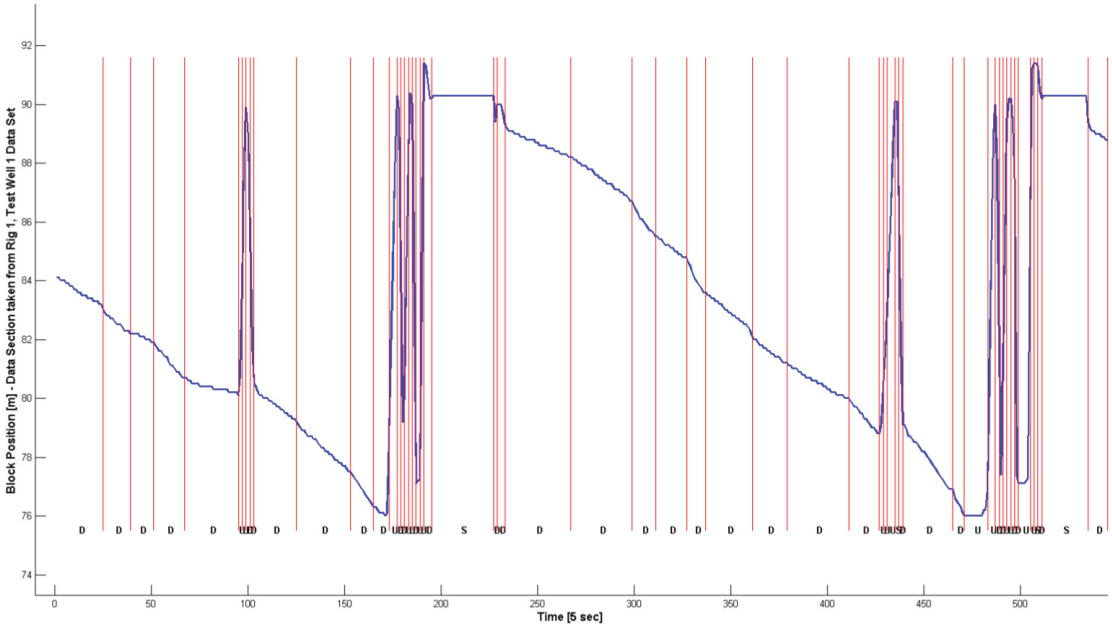


FIGURE 7.4: Piecewise Linear Approximation PLA, Bottom-Up approach applied on normal block position sensor data.

the rig performs other operations than drilling. PLA is applied on this data set and the results are plotted as vertical lines which represent the estimated segments and each segment is labeled with its trend (U, D, or S).

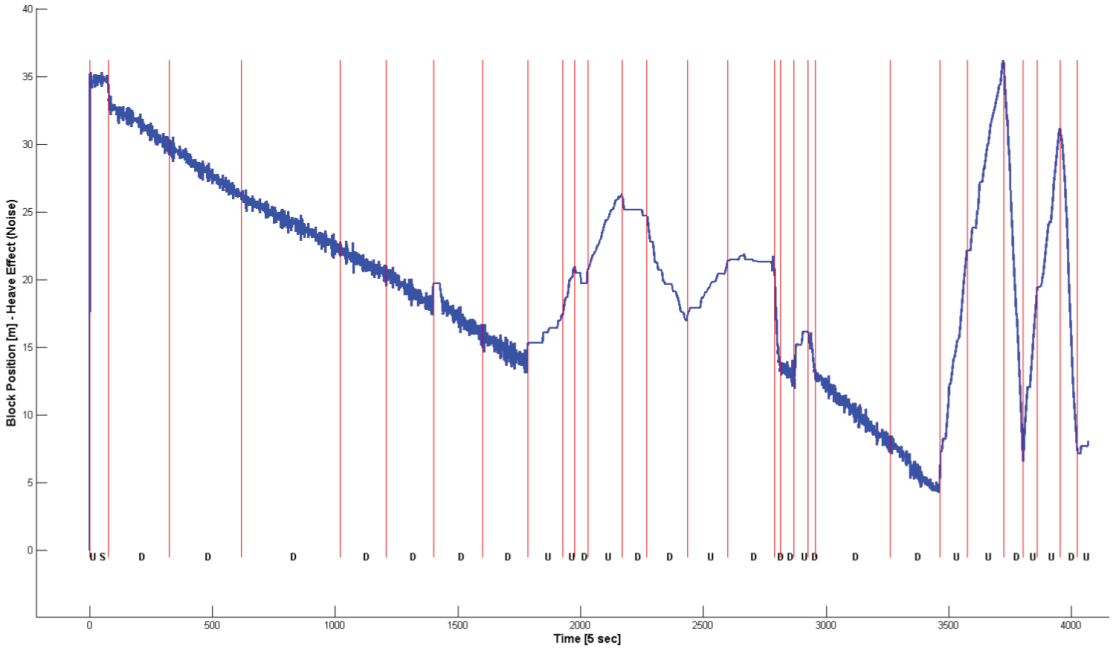


FIGURE 7.5: Piecewise Linear Approximation PLA, Bottom-Up approach applied on block position sensor data with heaves-compensation effect (common situation on drilling ships).

7.4.4 Joint Confidence Estimation of Piecewise Linear Approximation

During segments approximating using PLA, the cost of merging segment i with the segment $i+1$ is calculated. The distance between the merging cost and the maximum error represents the confidence estimation of joint between segments $\{i, i+1\}$. The bigger the distance, the more the confidence of estimation. The joint confidence is given by the formula:

$$joint\ confidence(i) = 100 - \frac{error_{max} * 100}{merge_{cost}(i, i+1)}. \tag{7.4}$$

Figure 7.6 shows block position sensor data segmented using PLA. On each segment joint, the confidence is estimated. It is clear that the second joint confidence estimation is 6% because the two segments have same trend (Down) and their slopes are very close to each other, while the next segments joint confidence is 96% where the slopes of segments are very far away from each other and first segment trend is (Down) and the second segment trend is (Up). The joint confidence can be used to refine the results of Piecewise Linear Approximation PLA algorithm where the segments with low joint confidence can be merged together if their trend is similar, this helps to granular the results and minimize the number of estimated segments.

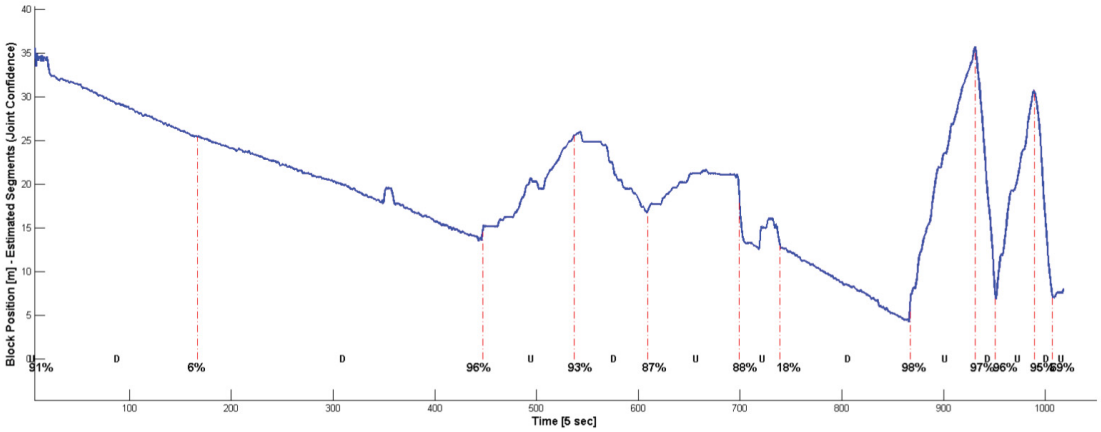


FIGURE 7.6: Piecewise Linear Approximation PLA, Bottom-Up approach, Confidence estimation of each segment's joint.

7.5 Summary

This chapter shows application of Piecewise Linear Approximation on block position to detect the states of drill-string movements. PLA shows simplicity in application and accuracy in results. The accuracy of results related heavily to $error_{max}$ parameter.

Filtering block position data could help the PLA algorithm to converge faster; also the results could be better because the filter could eliminate some erratic points that could affect the $merge_{cost}$ values and then the estimated segments will be changed.

Normalizing the data of block position helps in minimizing the changes to values of $error_{max}$ parameter when the PLA algorithm is applied on data sets from different rigs to get more accurate results.

By using clustering techniques (chapter 7), it will be possible to detect the drill-string states (InSlips/OutOfSlips), Pumps states (ON, OFF), and Rotary states (YES, NO). For each of those states, PLA is used to detect drill-string movements states (Up,Down, and Static). Those states will be used further in the suggested research hypothesis to detect the rig states from sensor data.

The author of this thesis discussed in detail the PLA algorithm and its application on the detecting of drill-string movements [68, 83].

Chapter 8

Rig State Detection using Shape Validation

8.1 Motivation

This chapter describes a novel algorithm to validate "InSlips" states and recognize "Make Connection/Disconnection" operations based on the shapes of hookload and block position sensor data.

"InSlips" is the state where the drill-string is hanged at rig floor and disconnected from rig's hook. In chapter 7, detailed description on detection of "InSlips" state is presented, a threshold-based approach is used to recognize "InSlips" statistically. The serious issue in this approach is that the boundaries of "InSlips" state are not accurately specified, the start and end time stamps refer to the hookload sensor data which is under the threshold (see figure 8.3), but according to drilling experts, this is a a serious issue in "InSlips" detection where the start of "InSlips" state should be located at the point at which drilling crew put the drill-string in slips. This point is normally located above the clusters threshold, because at this point the drill-string is not fully set in slips and the hookload values are relatively higher than the values when the drill-string is fully set in slips [70]. The same problem is at the end of "InSlips" state where the end point should be located at the point where the drill-string is fully out of slips, and also this point located above the calculated clusters threshold, figure 8.3 gives a good idea on this issue in detected "InSlips" states.

"Make Connection" is an operation performed by drilling crew during "InSlips" state to connect a new drill-stand to the drill-string [70]. This operation requires drill-string to be put in slips and disconnected from the hook, then the hook is connected to a

new drill-stand and carries it up to be able to screw it with the hanged drill-string. "Make Connection" operation is performed during running the drill-string in hole or during drilling operations. Another similar operation called "Make Disconnection", this operation occurs during "InSlips" state where the drill-stand is disconnected from drill-string. This operation is repeated during pulling drill-string out of hole [70].

In this chapter, Adjustment to start and end boundaries of "InSlips" state will be presented, then validation of "InSlips" is highlighted through validating the shape of hookload sensor data. Furthermore, shape validation of Block Position sensor data is used to detect the operations "Make Connection" and "Make Disconnection".

Gram polynomial moments are used as features to describe the shapes of hookload and block position sensor data, then a classifier is trained and tested on artificially-synthesized data set.

8.2 States as Shapes on Sensor Data

"InSlips" patterns on hookload sensor data has specific "U" shape. It started at the point where the drill-string is hung at hook and it has relatively a high value, then when drilling crew put drill-string in slips a jump down in the value appears at hookload sensor values, then the values of hookload show relatively steady state, then the drilling crew finish the work and the hook is connected again to drill-string, then drilling crew pulls the slips out and the hook carries all the weight of drill-string; this explains the high values of hookload sensor data.

Figure 8.1 demonstrates "InSlips" state with four different patterns on block position sensor data. The pattern 1 highlights "InSlips" state but block position sensor data shows low value at the beginning then it moves up and shows a high steady values. This represents a block position pattern during "Make Connection" operation where drilling crew puts drill-string "InSlips" then disconnects the hook from drill-string to connect another drill-stand, then it twists the drill-stand with drill-string. The shape of block position during "Make Connection" looks like "S" letter. The pattern 2 is similar to pattern 1, the different between them is that the pattern 1 happened between two drilling operations while the pattern 2 happens during running drill-string in hole. The pattern 3 shows an operation of disconnecting a drill-stand from drill-string, this operation normally happens during pulling drill-string out of hole, the shape of block position sensor data looks like "Z" letter. The final pattern 4 shows a steady block position where no movements happened to the travelling block, this means that the drilling crew put the drill-string in slips and performed another operation on the surface

or maybe they performed king of "flow check", sometimes, this requires other sensors to know what happened during this time slot.

During "InSlips", it is possible to have different block position shapes, this relates to the activity performed by drilling crew, they could put the drill-string in slips then they disconnect the hook and use it to perform some operations at rig's surface, then the shape of block position curve will be unpredictable.

Figure 8.1 summarizes the shapes of hookload and block position during different states: shape of hookload during any "InSlips" state should look like "U" letter; shape of block position during "Make Connection" should look like "S" letter; shape of block position during "Make Disconnection" should look like "Z" letter and any other shape of block position where the state is considered just "InSlips" but not any connection-related operation.

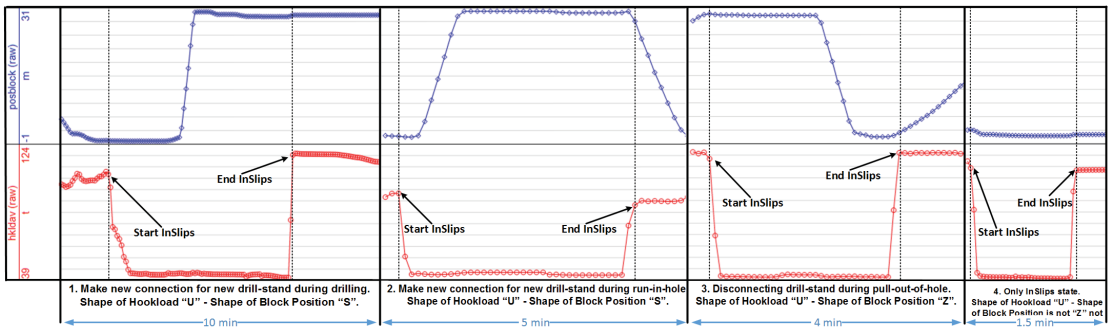


FIGURE 8.1: Shapes of Hookload and Block Position sensor data during different InSlips and Make Connection states.

8.3 Validation Process

In chapter 7, a development of an algorithm to detect "InSlips" on sensor data was presented, but as it is clear on figure 8.3 that the boundaries of detected "InSlips" are not correct, then it is required to put a validation process to correct the boundaries of "InSlips" states and validate the shapes of hookload and block position sensor data.

The validation process highlighted in figure 8.2 starts after performing clusters analysis and calculate the threshold to detect "InSlips" state, then the boundaries of "InSlips" states are corrected and adjusted, then the shapes of hookload sensor data are validated and compared against some pre-defined templates. The validated "InSlips" then transferred to validate the shape of block position sensor data against pre-defined templates of "Make Dis/Connection" operations. Before any comparison done to sensor shapes against templates, the shape descriptors or features will be calculated for each shape.

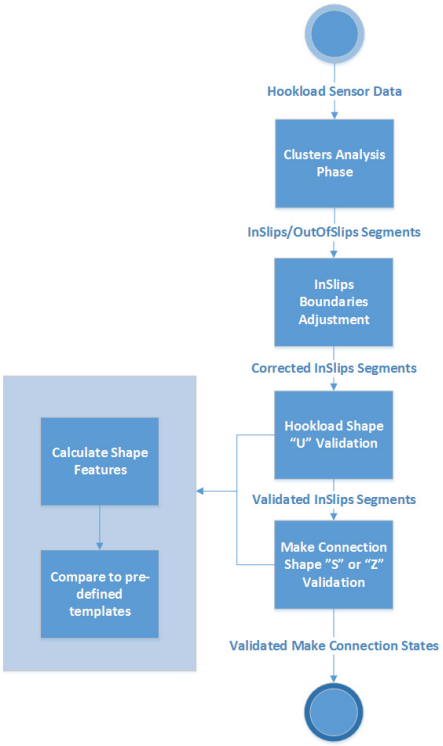


FIGURE 8.2: InSlips/MakeConnection Validation Process.

8.4 InSlips States Boundaries Adjustment

Figure 8.3 shows hookload and block position sensor data during detailed "InSlips" state and "Make Connection" operation. The detected segment of "InSlips" state is shorter than it should be, it is required to stretch it to the left and to the right in order to get correct "InSlips" segment. The correct start of "InSlips" state should be located at the first data point at hookload before it jumps down as effect of putting drill-string in slips. Similarly, the end index should be corrected and referred to the index of data point at which the drill-string is out of slips.

Figure 8.4 refers to the jumps in hookload data points between detected "InSlips" segment using cluster threshold and correct "InSlips" segment. It is obvious that those jumps in values are more than standard deviation of "InSlips" data segment, then the standard deviation of data segment during "InSlips" can be used as a border or a threshold to merge the data points to the segment. A searching process based on standard deviation value can be started after detection of "InSlips" segment to correct its start and end.

Table 8.1 contains the detailed algorithm for "InSlips" boundaries adjustment. It is simple iterative algorithm, it shows how the boundaries are corrected based on the jumps in data values.

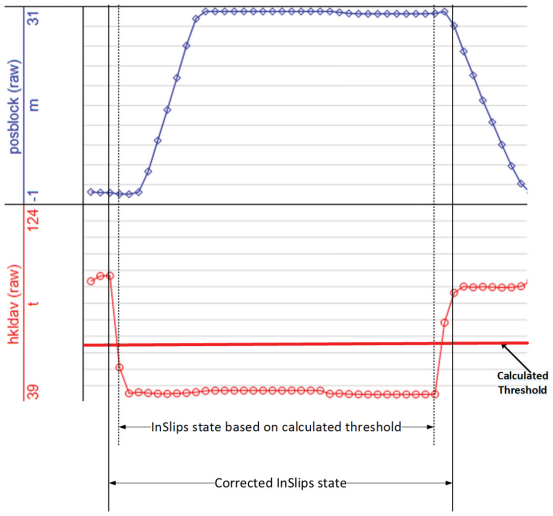


FIGURE 8.3: Boundaries of correct InSlips state.

FIGURE 8.4: InSlips Boundaries Adjustment.

8.5 Shape Features extraction

The Gram polynomial moments computed from sensor data can be used to identify the pattern (shape) of data. One of the method adopted to do moments based recognition is to compare the features - moments - vector calculated from given data segment to reference features vector [84–86].

In this work on sensors data, Gram polynomials are used to describe drilling time series during different states and operations. The choice of Gram polynomials is because not only their orthogonality but also those polynomials have a uniform scaling and this is important to fit some complex shapes in drilling time series with higher performance than what Vandermonde basis can do [87].

The recurrence equations for generating Gram polynomial is given by [87] as the following:

$$g_n(x) = 2\alpha_{n-1} x g_{n-1}(x) - \frac{\alpha_{n-1}}{\alpha_{n-2}} g_{n-2}(x). \tag{8.1}$$

InSlips Boundaries Adjustment Algorithm
Input: $Segment_{InSlips}$
Output: $CorrectedSegment_{InSlips}$
DO: **// Initiate Values** idxStart = $Segment_{InSlips}$.StartIndex; idxEnd = $Segment_{InSlips}$.EndIndex; dataSegment = data(idxStart:idxEnd); stdData = std(dataSegment); **// Correct Start Index** while (abs(data(idxStart) - data(idxStart - 1)) > stdData) idxStart = idxStart - 1; end; **// Correct End Index** while (abs(data(idxEnd) - data(idxEnd + 1)) > stdData) idxEnd = idxEnd + 1; end; **// Adjustment of segment boundaries** $CorrectedSegment_{InSlips}$.StartIndex = idxStart; $CorrectedSegment_{InSlips}$.EndIndex = idxEnd;

TABLE 8.1: Adjustment of InSlips Boundaries Algorithm

Whereby,

$$\alpha_{n-1} = \frac{m}{n}\left(\frac{n^2 - \frac{1}{2}}{m^2 - n^2}\right)^2, \tag{8.2}$$

and

$$g_0(x) = 1, \qquad g_{-1}(x) = 0 \qquad and \qquad \alpha_{-1} = 1. \tag{8.3}$$

Figure 8.5 shows two data segments of hookload sensor data with their Gram polynomial spectrum and total proportional power of the spectrum. The hookload sensor data and their estimation is plotted on same area (at figure 8.5, raw data: blue line, estimated data: red line). The Gram polynomial spectrums of two "InSlips" hookload segments look similar to each other, this supports the idea of using reference or template of features vector to validate all new data segments against it. The degree 20 of Gram polynomials is used to approximate the raw data. The question that could arise here is what is the best degree that can be used to represent the raw data of "InSlips" state? The total proportional power can help in answering this question. It shows the accumulated information carried by adding new degree or component to Gram polynomials, it shows on both plots of total proportional power of two hookload data segments that the degree of 10 is the optimal degree (horizontal red line is plotted at level 95%) i.e. using Gram polynomials of degree 10 is enough to represent the 95% of raw hookload data segments.

The degree of polynomial can be adopted dynamically. The level of proportional power can be fixed on a predefined level (for example 95%). This level can be chosen by experts to see how good the matching of the real data with the template. Then the degree of polynomial can be determined dynamically using the method described in the figure 8.6.

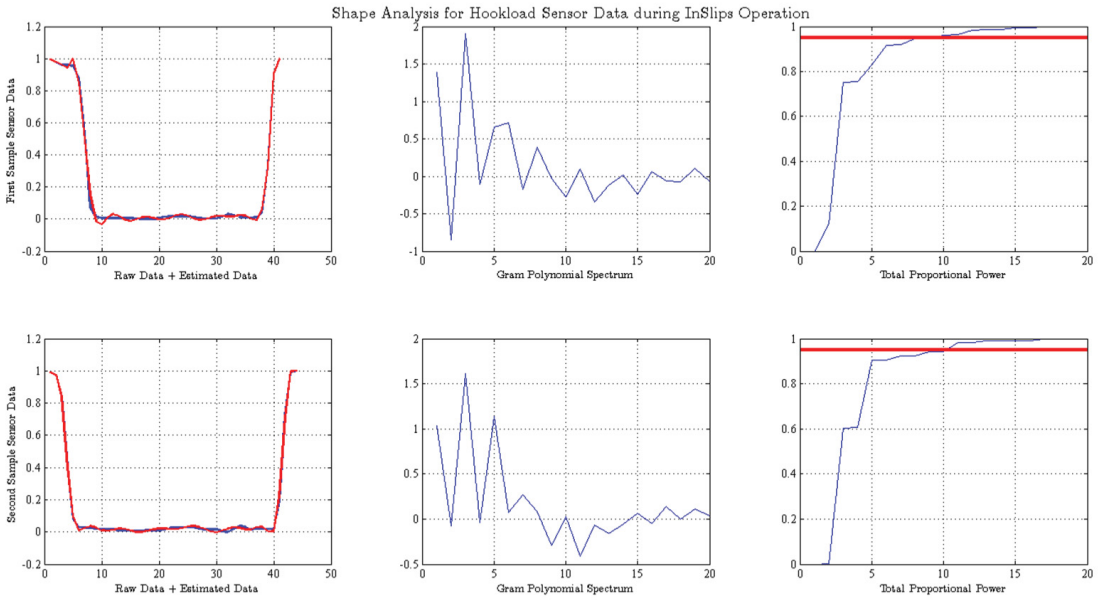


FIGURE 8.5: Polynomial moments as features in shape of hookload sensor data dusing InSlips state - Shape "U".

Two block position sensor data segments for operation "Make Connection" are analyzed in figure 8.6. The analysis results show that the both data segments have similar polynomial spectrum or features vector. The total proportional powers indicate to degree 5 as best Gram polynomial degree to reconstruct the raw data.

Similar analogy is performed to two data segments of block position sensor data during operations of "Make Disconnection" (see figure 8.7'). Also here, there is a similarity in polynomial spectrum or features vector. The best degree of Gram polynomials to represent the raw data is degree 5 as the level of 95% indicates on the total proportional power (red line).

8.6 Shape Classification

The objective of a classifier is to identify whether the unknown input data segment has specific valid shape or not. Euclidean distance measure is used to do the classification. The minimum distance classifier is used to classify the unknown sensor data segment from the testing set according to templates (reference segments). The templates of reference segments were chosen by drilling experts. During the classification process,

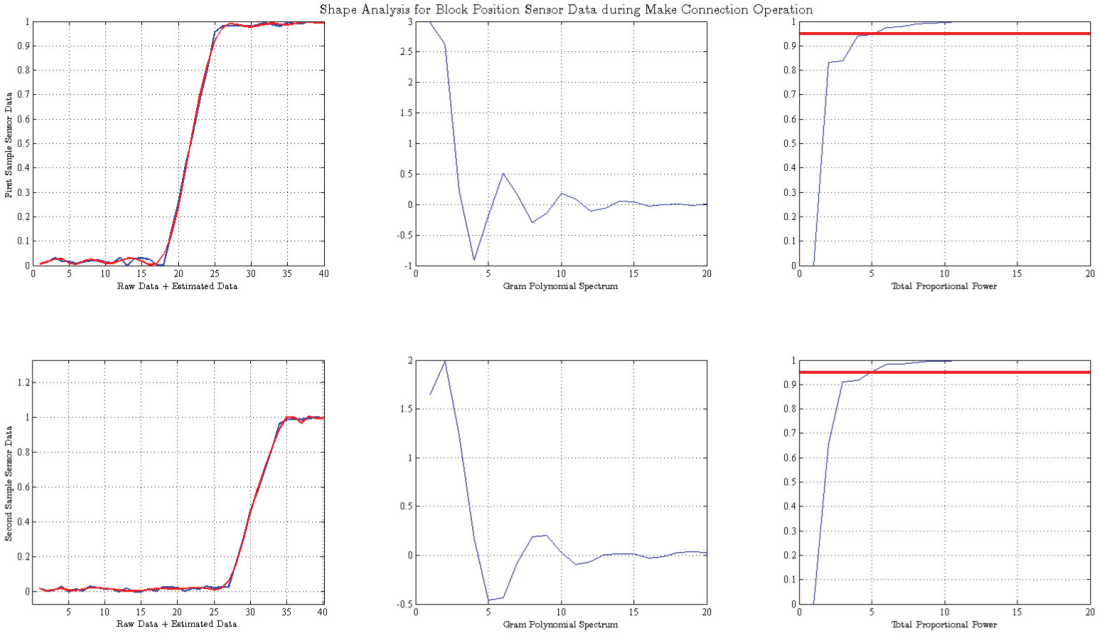


FIGURE 8.6: Polynomial moments as features in shape of block position sensor data during Make Connection- Shape "S".

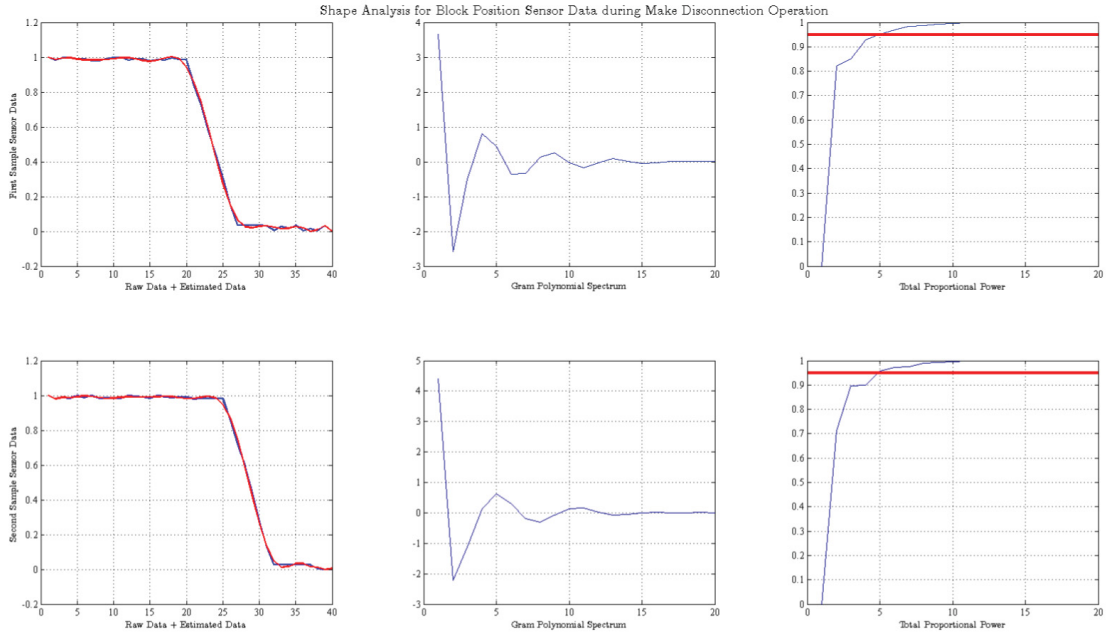


FIGURE 8.7: Polynomial moments as features in shape of block position sensor data during Make Disconnection- Shape "Z".

features of the unknown data segment are computed and then compared to reference feature vectors before a decision on shape validation is issued. The Euclidean distance measure is commonly used for classification purpose [88] and it is given by:

$$D = \sqrt{\sum_{i=1}^{d} (V_{template} - V_{input})^2}, \tag{8.4}$$

where d is the degree of Gram polynomials, $V_{template}$ is reference features vector which contains the coefficients of Gram polynomial moments and V_{input} is the feature vector of input data segment.

8.6.1 Recognition Accuracy

The recognition accuracy estimation is used as evaluation criteria. The recognition accuracy γ of n samples is defined by [89]:

$$\gamma_n = \frac{C_s}{T_s}, \tag{8.5}$$

where C_s is number of correctly classified segments and T_s is total number of segments.

8.6.2 Confusion Matrix

Confusion matrix is a technique used to compare the results of classification. It is a table where the samples based on their actual classes displayed as rows and the prediction results displayed as columns, each cell in the table contains the number of predicted instances from column class. The precision of classifier for a given class is the total number of correctly classified instances over the total number of instances from the given class. Recall of classifier is the number of correctly classified instances over the number of all instances predicted as the given class. The diagonal elements represent the correctly classified instances [81].

The total accuracy of classification system can be calculated from the confusion matrix by [89]:

$$Accuracy_{total} = \frac{\sum_{i=1}^{k} P_i}{\sum_{i=1}^{k} T_i}, \tag{8.6}$$

where k is the number of predicted classes, P_i is the number of correctly classified samples of class i, and T_i is the total number of actual samples of class i.

8.7 Experimental Results

To prove the concept of shape detection on sensor data, a testing data set of shapes “U”, “S” and “Z” is generated. The data set contains 4500 samples each of which represents different shape, the data set is divided on the shapes where each shape type has 1500

samples. Drilling experts supported choice correct reference templates for each shape from real data as well as helping in generating the shapes randomly using a computer program. Then a classification process is executed on the testing data set and the results showed in confusion matrix.

Figure 8.8 shows a part of "InSlips" states - shape "U" - in the testing data set, another section of testing data set for shapes "S" highlighted in figure 8.9, while figure 8.10 contains 200 samples of "Z" shapes each of which is generated randomly.

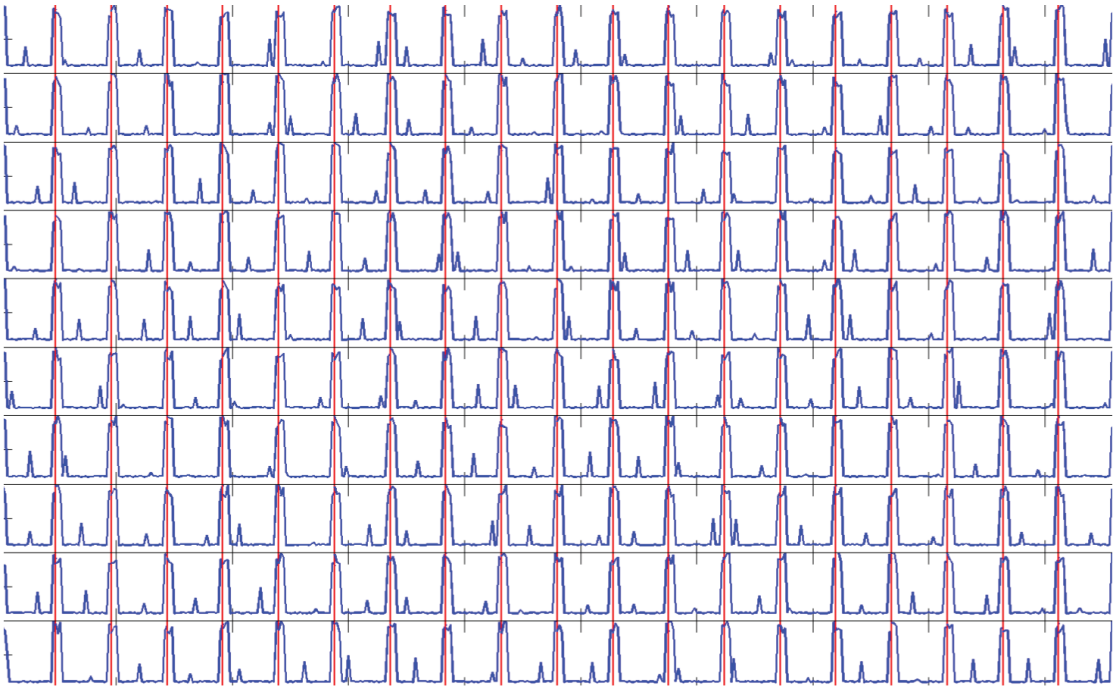


FIGURE 8.8: Hookload sensor data of 10x20 InSlips - "U" shape - from Artificially-Generated Test Data Set.

The results of classification process on testing data set of 4500 samples displayed in figure 8.11. Those results obtained after careful choice of the euclidean distance threshold that is used by the classifier to determine whether an input segment has specific shape class. The total accuracy of the classifier is 93.42% which is a good accuracy, also the classifiers has very good precision and recall.

The percentage 93% of the classification shows that the classifier failed in a number of cases. Those cases will be discussed in the next paragraph "Borderline Cases".

8.7.1 Borderline Cases

The figure 8.12 demonstrates 8 cases of different shapes from "InSlips" class. Each case (blue line) is matched with a template (red line) chosen by a drilling expert. The first case is a normal case of "InSlips" where the distance is 1.96 and it shows a good

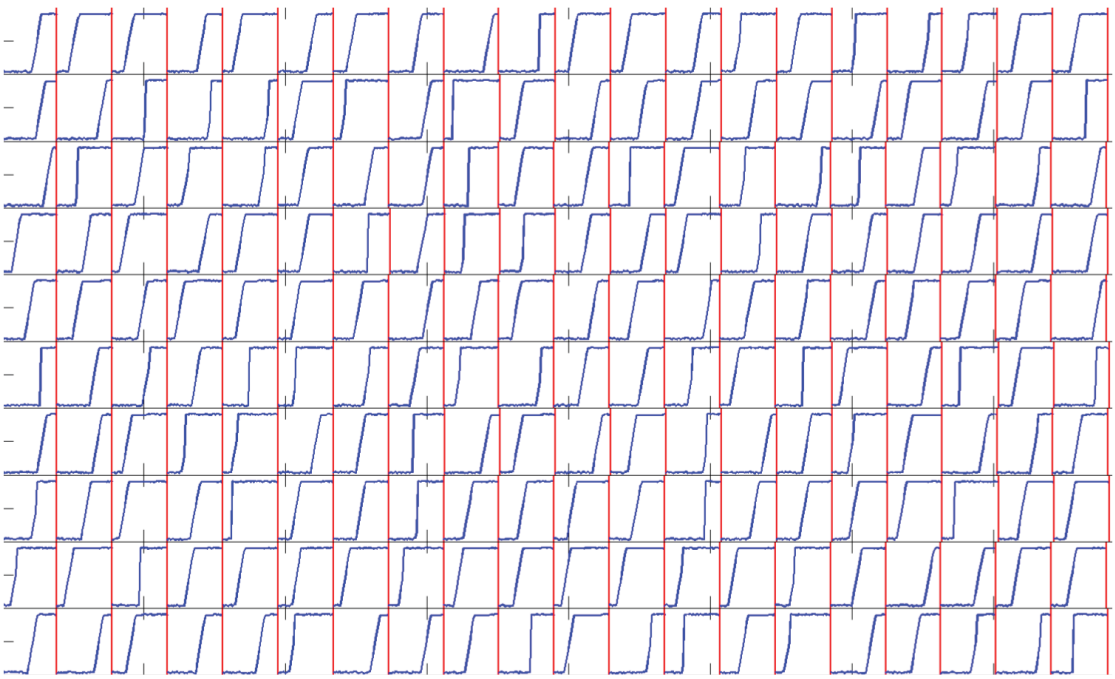


FIGURE 8.9: Block position sensor data of 10x20 Make Connections - "S" shape - from Artificially-Generated Test Data Set.

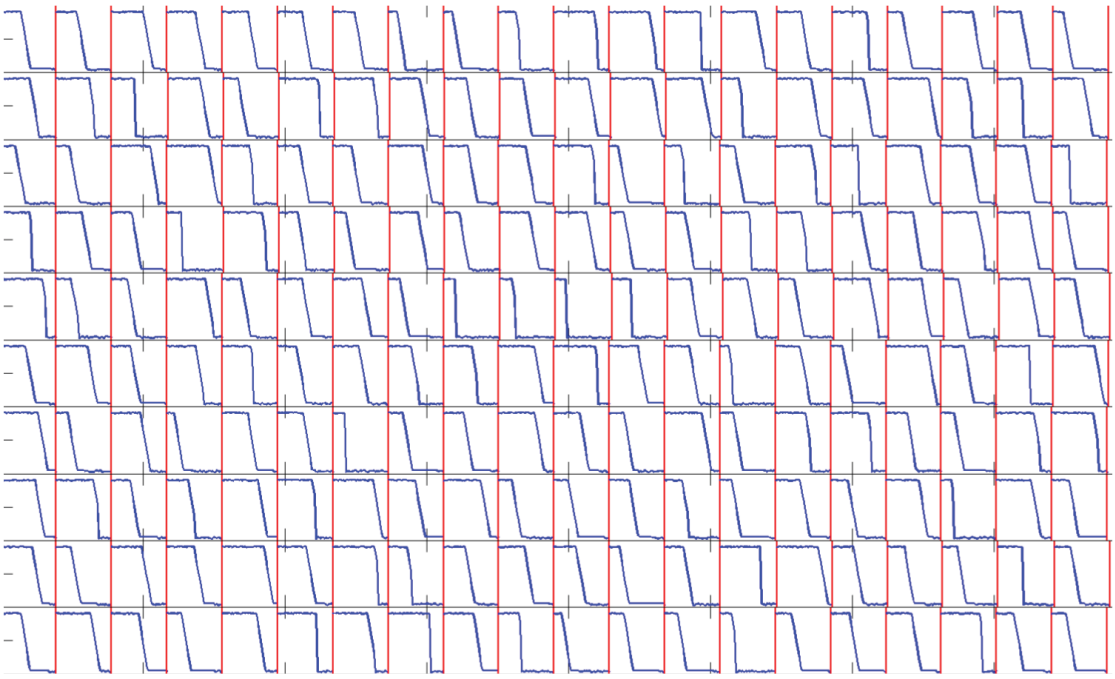


FIGURE 8.10: Block position sensor data of 10x20 Make Disconnections - "Z" shape - from Artificially-Generated Test Data Set.

match between the raw data and the template. The second case is a long "InSlips" which is around three times longer than the first case. This case shows also a good match with the template with distance 1.73. The third case is very long "InSlips" and here it is obvious that distance is almost two times as the previous case. This is due to the approximation of the long data using the Gram Polynomials i.e. approximating

		Predicted			
		"U" Shape	"S" Shape	"Z" Shape	Precision
Actual	"U" Shape	1389	26	85	92.60%
	"S" Shape	45	1434	21	95.60%
	"Z" Shape	41	78	1381	92.07%
	Recall	94.17%	93.24%	92.87%	
		Total Accuracy		93.42%	

FIGURE 8.11: Confusion Matrix of classification results of three shape types "U", "S" and "Z".

very long data required higher components with higher magnitude in the polynomial spectrum. The case 4 shows a noisy "InSlips" data, it seems that the suggested method has a good tolerance to the spikes that are in a positive (up) direction. The case 5 is a case of "InSlips" with spike in positive direction and another spike in negative direction. The distance shows that this case is not matching good with the template. This is because of the normalization process. The normalized data has different magnitude of the pattern parts than the template. And this makes the approximation process picking higher or lower magnitudes of the components in polynomial spectrum to be able to approximate the data. The case 6 confirms the conclusion in case 5. The negative spike affects the normalization process. And then the distance to the template is bigger than the distance in the case 4 where there is just one positive spike. The case 7 is a case of short "InSlips". The case 8 is just a random shape with one up step. The distance in this case is relativelly a bigger than all other steps. It is around 4.2. As a conclusion, by picking a suitable distance, it will be possible to detect all "InSlips" cases and isolate the non "InSlips" cases. The distance picking process should be performed by consulting drilling experts to determine which cases should be recognized as "InSlips".

8.8 Summary

In this chapter, a validation process of "InSlips" and "Make Dis/Connection" operations was presented. The validation process was required due to the limitation of detected "InSlips" segments using clusters analysis technique. The focus of this chapter was on how to improve the "InSlips" segments obtained from clustering analysis of hookload sensor data using boundaries adjustment algorithms. The "InSlips" boundaries adjustment algorithm is discussed in detail with its pseudo code. A classifier was developed to perform the main step in validation process, it is trained to classify the shapes of sensor data segments into three shape types "U", "S" and "Z". The high accuracy of classification supports the idea of using this approach in shape validation process.

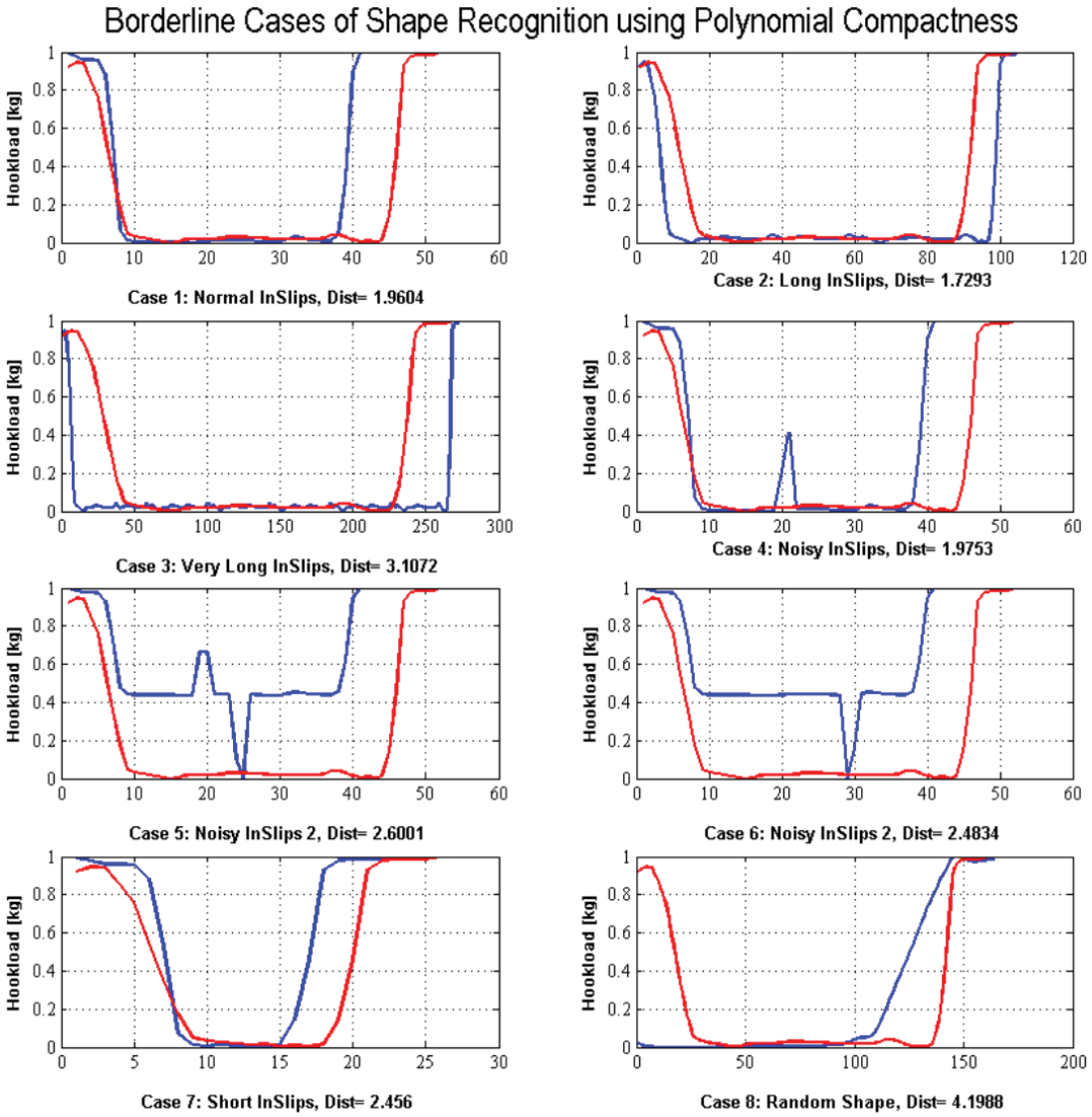


FIGURE 8.12: Borderline cases of Shape Recognition using Polynomial Compactness (Raw Data is the blue line, The template is the red line).

Further steps can be applied to improve the results more, one of those step is the usage of another type of classification techniques such as Support Vector Machines, Rules Induction, K-NN, or Neural Networks to recognize shape types of input data segments. Another enhancement for the suggested work by measuring the stability of each applied classier by adding noise with different degrees to the randomly-generated testing data set.

Chapter 9

Distributed State Detection System

9.1 Motivation

This chapter deals with detection of rig state based on state of rig's sub-systems. In previous chapters 6-7-8, the states of rig's sub-systems were detected based on the sensors data, chapter 6 shows how to get the state of drill-string (InSlips/OutOfSlips) from hookload sensor data, state of pumps (On/Off) from flowrate sensor data, and states of rotary system (Yes/No) based on RPM sensor data. Chapter 7 presents detection of drill-staring movements (Up/Down/ Static) based on block position sensor data. Chapter 8 demonstrates how to correct "InSlips" and recognized "Make Dis/Connection".

In this chapter, general framework for merging and fusing the information from different sensors will be suggested, it starts with review of rig state knowledge and the relationship between sensor states and rig states, it then suggests a distributed fusion model for sensor data. The suggested framework is distributed over components that communicates through middleware. Results of rig state detection by implementing the distributed system are presented at the end of this chapter.

9.2 Rig State Knowledge

The table in figure 9.1 helps to understand the relationship between rig state and sensors states. The rig state is a combination of other states of rig's sub-systems, for example to consider that a rig in "Drilling Rotary" state, it is required to have RPM as "YES",

flowrate as "ON", drill-string as "OutOfSlips", drill-string as moving "Down", and drill-bit as "OnBottom" of hole. State of "OnBottom" means that the bit is close to the bottom of hole, it is a trivial task to consider the bit is touching the bottom of hole by calculating the difference between depth of hole and depth of bit. Through a provided threshold from drilling experts, these states can be easily recognized. It is clear that states of "InSlips" and "Make Connection" are related directly to the "InSlips" state of hookload where a shape validation process will be executed to recognize exact "InSlips" and "Make Connection" based on the shapes of hookload and block position sensors data (see chapter 8).

Based on information presented in figure 9.1, it is possible now to conclude all the rig states based on the states of sensor data. The next paragraphs of this chapter discuss how to merge the information from different sensors in order to give a decision on the rig state, also a discussion on getting the sensor data from rig site through WITSML Bridge Broker is highlighted.

Sensor State	Rotation (RPM)	Flowrate	Bit Movement			Block Position Movement			Hole Depth Increase	On Bottom (Yes, No)	Hookload	
Rig State			UP	DOWN	Static	UP	DOWN	Static			OutOfSlips	InSlips
Drilling rotating	Yes	ON	No	Yes	No	No	Yes	No	Yes	Yes	Yes	No
Drilling sliding	No	ON	No	Yes	No	No	Yes	No	Yes	Yes	Yes	No
Wash downwards (into hole)	No	ON	No	Yes	No	No	Yes	No	No	No	Yes	No
Wash upwards (out of hole)	No	ON	Yes	No	No	Yes	No	No	No	No	Yes	No
Ream downwards (into hole)	Yes	ON	No	Yes	No	No	Yes	No	No	No	Yes	No
Ream upwards (backreaming)	Yes	ON	Yes	No	No	Yes	No	No	No	No	Yes	No
Run in hole	No	OFF	No	Yes	No	No	Yes	No	No	No	Yes	No
Pull out of hole	No	OFF	Yes	No	No	Yes	No	No	No	No	Yes	No
Make connection	-	-	-	-	-	-	-	-	-	-	No	Yes
In Slips	-	-	-	-	-	-	-	-	-	-	No	Yes
Circulate hole	-	ON	No	No	Yes	No	No	Yes	No	No	Yes	No

FIGURE 9.1: Rig State Knowledge - Relationship between sensor states and rig states.

9.3 Distributed Multi-sensor Fusion Model

In order to perform the last step in this thesis, it is required to fuse all the information from rig sensor data together. Figure 9.2 shows the suggested fusion model using which the states from sensors can be calculated and merged, and then the decision will be issued on rig state. This fusion model is a mixture of two data fusion models: Waterfall Model [22] and Distributed Blackboard Model [20]. The choice of Waterfall model because the steps of fusion process are essential in rig state detection case, where the

sensors data need to be preprocessed and filtered before the features on states of machines can be concluded, then rig's sub-systems can be recognized so the final step will be making decision on rig's state. The idea of using supervisory components for each sensor is used from Distributed Blackboard Fusion model. The supervisory components perform pre-processing on sensors data. The pre-processing phase is applied on each sensor data to get them filtered form outliers and unwanted noise. An lowpass Chebyshev Type I filter with a cutoff frequency of $0.8 * (Fs/2)/r$ is applied where Fs is sampling rate and r is the down sampling factor [90].

Features extraction phase is located after preprocessing phase, the idea here is to extract the states from sensors data and consider them as features, for example: drill-string state (InSlips/OutOfSlips) from hookload sensor data, drill-string movements (UP/-Down/Static) from block position sensor data, pumps state (ON/OFF) from flowrate sensor data, drill-string rotary state (YES/NO) from RPM sensor data, Bit state "On-Bottom" (YES/NO) from mdhole and mdbit sensor data.

Rig's Sub-System Situation Assessment is a fusion phase in which the states of hoisting system (Slips state/Moving state/Bit state), rotary System (Rotating), and circulation system (Pumping) are evaluated and integrated.

Decision Making is a phase to fuse the states of rig's sub-systems and machines in order to estimate the state of the rig. The feedback line is important because it carries information on processing parameters of state detection algorithms after estimating results by drilling experts.

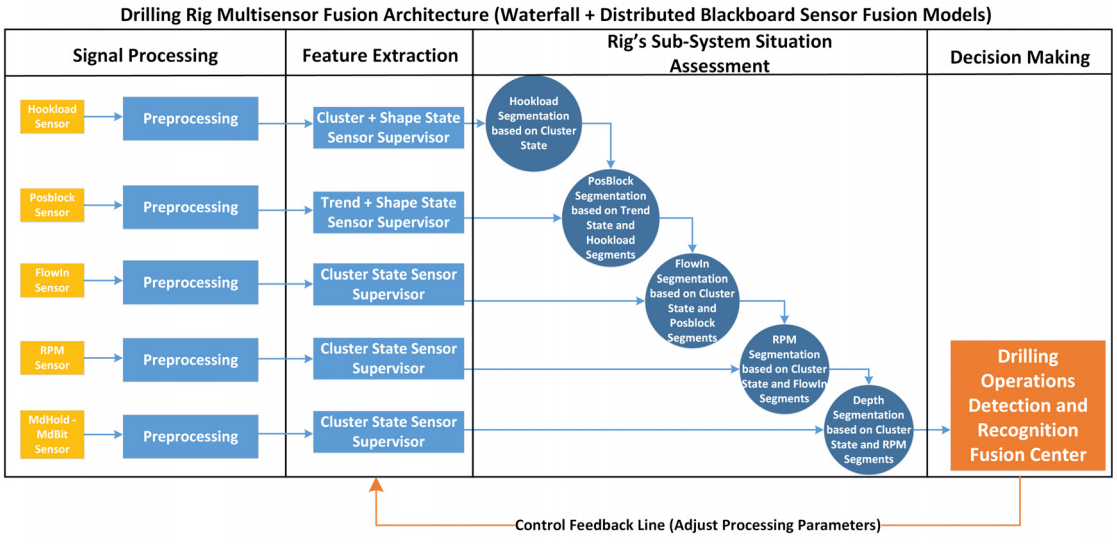


FIGURE 9.2: Distributed Sensor Fusion Model

The Waterfall model is shown during the Rig's Sub-System Situation Assessment. The segmentation of Drill-String Movements (Posblock) is used based on the segments Hookload (Drill-String is "InSlips" or "OutOfSlips". Also the segmentation of flowin based on posblock segments. The RPM segments are calculated based on flowin segments. The Depth segmentation (Drill-bit touches the bottom of hole) is based on RPM segments. Finally all the segments will be fed to the Decision Making Phase.

9.4 Distributed System Architecture

Figure 9.3 suggests distributed architecture of rig's detection system. The architecture is distributed over many processing components each of which hosts an algorithm for sensor data processing.

At rig site, the data is collected from sensors through a measurement system via rig's Service Company, then the sensor data is stored temporarily at WITSML server to be transferred to office site or any interesting party (see chapter A for more information). At office site a WITSML Client is used to acquire the data via a distributed model for data distribution, this model called Publish/Subscribe model where the WITSML client broker is sending a subscription with the sensor data to WITSML Server, and when this data is available then the WITSML server publishes this data back to the client. Once the data is received by WITSML Client, the client publishes it again to internal middleware to make it available for all other interested components.

Preprocessing Component receives the raw data and filter it and publishes it back to the middelware. Hookload segmentation component receives the raw data of hookload sensor and it does segmentation over the data and publishes the segments back to middelware. Block Position segmentation component get the hookload segments and for each Out-OfSlips segments, it slices it into segments based on the movements of drill-string. The RPM segmentation receives the block position segments and it does segmentation to them into small segments based on drill-string rotational state, then it publishes those segments back to middelware. Flowrate segmentation component receives the rotational segments from middelware, it segments them into smaller segments based on pumps's state, then it publishes back the segments to the middelware. Depths segmentation components reads the flow segments from middleware and then it segments them into more smaller segments based on bit state and publishes them back to middleware. The component of rig state classification receives all the segments from all components and then it takes a decision on what the states of the rig for each received segments are.

In this case, the middleware plays the role of the heart in communicating process between the components. The real advantage of using middleware in the suggested architecture is that the components are loosly-coupled which means that the components are not depending on each other to perform their work i.e. there is no need for the components to wait till the others finish their jobs, this gives the system high performance rate and tolerance against any failure in any processing component.

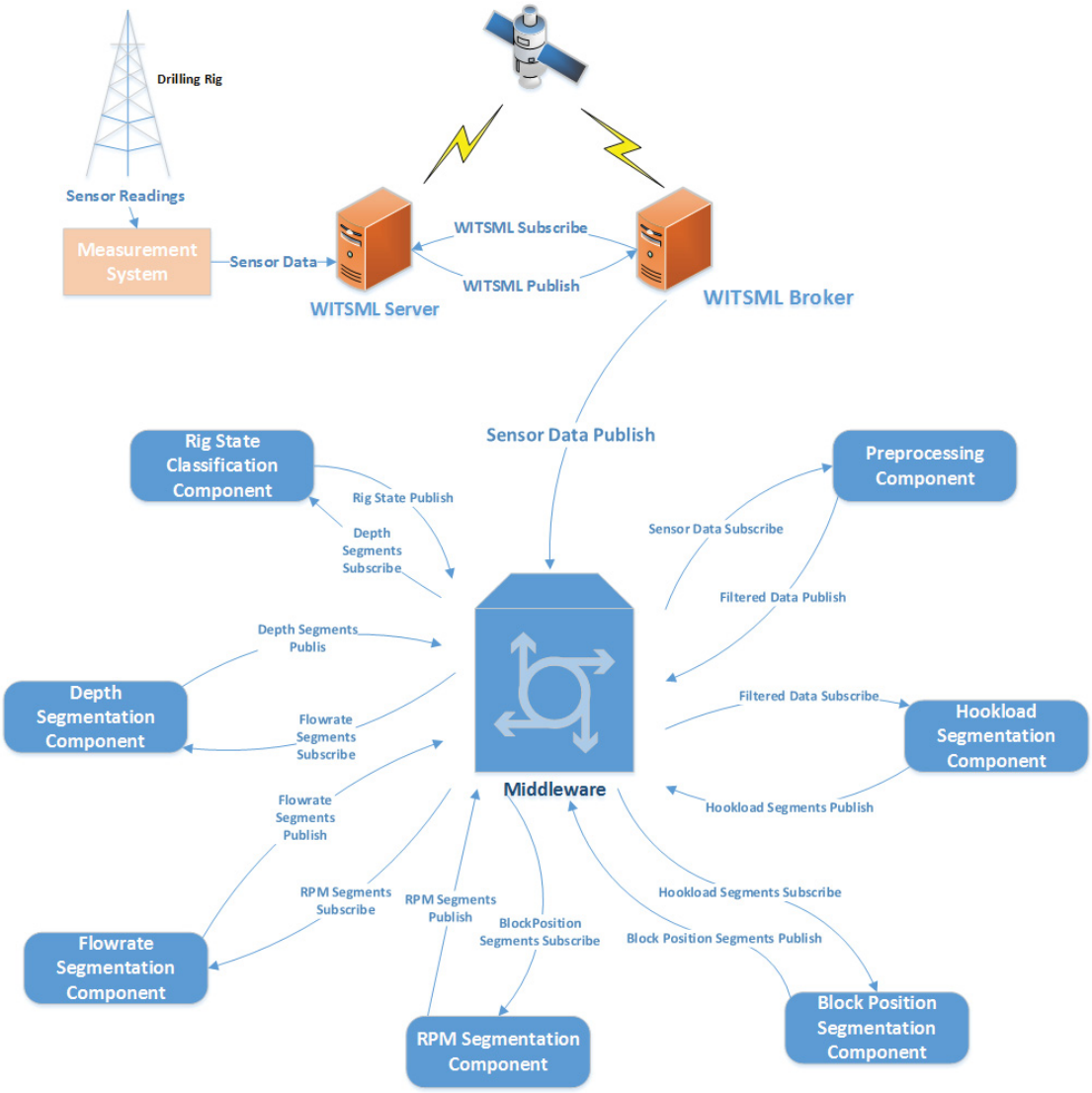


FIGURE 9.3: Distributed System Architecture using Middleware and Publish/Subscribe Model

9.5 Rig State Detection Process

The algorithm presented in table 9.1 shows how the sequence of execution from the reading phase to decision making and rig state detection phase. It starts with pre-processing and filtering of all sensors data, then the first phase is hookload segmentation to detect (InSlips/OutOfSlips) segments, then for each "InSlips" segment, it is processed through adjustment of segment's boundaries and then validated the hookload sensor data shape. Shape validation is performed also on block position sensor data during "InSlips" segment to recognize "Make Dis/Connection" operations. If the segment is "OutOfSlips" then the processing goes through block position sensor data segmentation to get the segments (Up/Down/Static), then the RPM segmentation process will be started to segment block position segmentation into smaller segments with rotational (Yes/No) segments. Flowrate segmentation is started to segment the rpm segments into (On/Off) segments, then depths segmentation process slices flow segments into bit "OnBottom" Yes/No segments. The rig states classifier is the component that constructs a decision tree based on the table 9.1 to classify each obtained segment as rig state.

9.6 Experimental Results

Test data Set	Duration, days	Sampling Int, Sec	Data Points, Count
TD92	72	60	103,680.00
TD1246	13.6	1	1,175,040.00
TD1258	11.7	1	1,010,880.00
TD1203	28.3	1	2,445,120.00
TD969	42.5	5	734,400.00
TD987	28.1	4	606,960.00
TD56	7.2	10	62,208.00
TD287	3.8	10	32,832.00
TD285	5	10	43,200.00
TD101	65	10	561,600.00
TD5	53	10	457,920.00
Sum, days:	330.2	Sum, Count:	7,233,840.00

FIGURE 9.4: Test Data Sets

Eleven test data sets of sensor data were collected from drilling rigs from different parts in the world using WITSML Bridge Broker. The sensors are: Hookload, Block Position, RPM, FlowIn Rate, Hole Depth, Bit Depth. All the data sets were classified manually by drilling experts in cooperation with information from daily drilling reports. Figure 9.4 shows information on each test data set. Each of these data sets is classified using the suggested prototype to detect and recognize the states of rigs. Furthermore, each data sets is classified using different processing parameters. More than 330 days of drilling (7.2 million sensor data points) fetched from rigs sites and processed using this

Rig State Detection Algorithm

Input:
SensorData

Output:
States

DO:
// Preprocessing All Sensor Data
FilteredData = Filter(*SensorData*);
// Hookload Segmentation
HSegments = SegmentationUsingEnvelope(FilteredData);
// Process only OutOfSlips Segments
for i = 1 : length(*HSegments*)
if (HSegments(i) has *InSlips* state) {
// Validate InSlips Shapes
ValidatedInSlipsSegments =
ShapeValidate(FilteredData(HSegment(i).Range,'Hookload'),
InSlipsTemplate);
// Validate Make Dis/Connection
ValidatedConnectionSegments =
ShapeValidate(FilteredData(HSegment(i).Range,'Posblock'),
MakeDis/connectionTemplate);
}else{
// Block Position Segmentation
StringMoveStateSegments = PLA(FilteredData(HSegment(i).Range,'Posblock'));
// RPM Segmentation
RPMStateSegments =
SegmentationUsingEnvelope(FilteredData(HSegment(i).Range,'RPM'),
StringMoveStateSegments);
// Flowin Segmentation
FlowinStateSegments =
SegmentationUsingEnvelope(FilteredData(HSegment(i).Range,'Flowin'),
RPMStateSegments);
// Depth Segmentation
DepthStateSegments =
SegmentationUsingEnvelope(FilteredData(HSegment(i).Range,'Mdhole',
'MdBit'), FlowinStateSegments);
}
end
// Classify all the rig states
for i = 1 : length(*DepthStateSegments*)
States(i) = ClassifySegment(DepthStateSegments(i));
end

Table 9.1: Rig State Detection Algorithm - Sensor Data Processing Algorithm

prototype. Figures 9.6-9.7-9.8-9.9 illustrate the confusions matrices of states detection process. The figure 9.5 shows a sketch of raw sensor data with results of segmentation and classification process. The segments are displayed by vertical red lines, and the rig states are showed using colored rectangles.

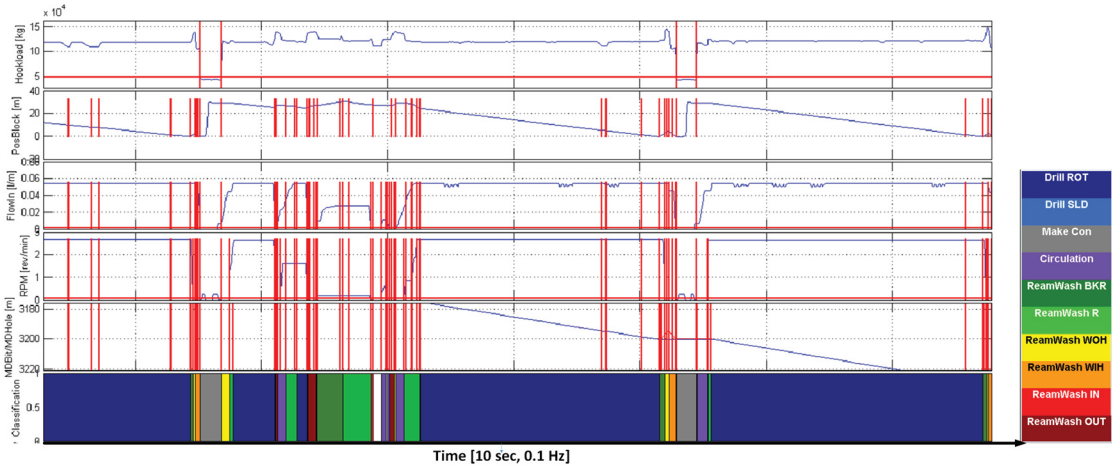


FIGURE 9.5: Results of Rig States Detection - Detailed View

In the next paragraphs, detailed discussion and general conclusions on each confusion matrix will be summarized.

9.6.1 Test Rigs: TD92, TD1246 and TD1258

Figure 9.6 shows the confusion matrices of rig state detection of **TD92, TD1246** and **TD1258**. The total accuracy is located between 92% and 96% which is considered as high accuracy. In **TD92**, the actual "Drilling ROT" sometimes predicted by the suggested system as "Reaming" state, because that the reaming state can be performed when the drill bit is close to bottom of hole, and this means that "OnBottom" state can be "Yes" and then the state is recognized as "Drilling ROT" instead of "Reaming". The same situation existed on data sets **TD1246** and **TD1258**, especially that directions of block position at both states are the same. Another reason for this can be issues related to quality of received sensor data. "Make Dis/Connection" show a very high accuracy around 99% because that this state is mainly related to "InSlips" state, the false detected "Make Dis/Connection" is due to the issue due bad detected "InSlips" at the end of tripping drill-string out of hole where the value of hookload sensor data at "InSlips" is very close to the value at "OutOfSlips". In data sets **TD92, TD1246** and **TD1258**, it is shown that there is no confusion between "Reaming" and "Back Reaming" and this is due to opposite directions of Travelling Block which is detected using Picesewise Linear Approximation algorithm, the same case is between "Run In Hole" and "Pull Out Of

Hole" states. Another issue is related to the confusion of circulation states because of missing calibration of block position changes in PLA algorithm.

Test Data Set: TD92

Total Accuracy: 92.69%

Actual Rig States \ Detected Rig States	Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Drilling ROT	**123**	2	0	13	0	6	0	**85.42%**
Run in Hole	**3**	**105**	0	9	0	3	5	**84.00%**
Pull out of Hole	**0**	0	**95**	0	16	5	2	**80.51%**
Reaming	**2**	12	0	**221**	**0**	6	0	**91.70%**
Back Reaming	**0**	0	5	5	**217**	2	0	**94.76%**
Circulation	**1**	5	4	6	15	**113**	0	**78.47%**
MakeCon	**0**	0	3	0	0	0	**775**	**99.61%**
Recall	**95.35%**	**84.68%**	**88.79%**	**87.01%**	**87.50%**	**83.70%**	**99.10%**	**92.69%**

Test Data Set: TD1246

Total Accuracy: 95.20%

Actual Rig States \ Detected Rig States	Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Drilling ROT	**210**	8	0	17	0	5	0	87.50%
Run in Hole	**5**	**741**	0	8	0	2	12	96.48%
Pull out of Hole	**0**	0	**535**	**0**	13	4	4	96.22%
Reaming	**2**	17	0	**298**	0	3	0	93.13%
Back Reaming	**0**	0	13	0	**118**	9	0	84.29%
Circulation	**3**	7	5	0	12	**381**	0	93.38%
MakeCon	**0**	0	2	3	1	0	**789**	99.25%
Recall	**95.45%**	**95.86%**	**96.40%**	**91.41%**	**81.94%**	**94.31%**	**98.01%**	**95.20%**

Test Data Set: TD1258

Total Accuracy: 95.59%

Actual Rig States \ Detected Rig States	Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Drilling ROT	**250**	6	0	12	0	1	0	92.94%
Run in Hole	**6**	**589**	0	14	0	2	2	96.08%
Pull out of Hole	**0**	2	**481**	**0**	**9**	0	6	96.59%
Reaming	**2**	21	0	**299**	**0**	1	0	92.57%
Back Reaming	**0**	0	23	**0**	**238**	4	0	89.81%
Circulation	**9**	9	0	3	4	**511**	0	95.34%
MakeCon	**1**	0	0	4	1	3	**774**	98.85%
Recall	**93.28%**	**93.94%**	**95.44%**	**90.06%**	**94.44%**	**97.89%**	**98.98%**	**95.59%**

FIGURE 9.6: Confusion Matrices of TD92 - TD1246 - TD1258

9.6.2 Test Rigs: TD1203, TD969 and TD987

In Figure 9.7, another data sets **TD1203, TD969** and **TD987** are presented with accuracy range $[97\%, 99\%]$. "Drilling ROT" state is confused more with "Run In Hole" and "Circulation" and sometimes with "Make Connection", the confusion with "Run in Hole" is due to some lowest values during drilling under the level of hookload at "InSlips", in some drilling ships, the case of low hookload during drilling is possible, the confusion with "Circulation" due to calibration issue in PLA algorithm because

PLA is not sensitive enough to detect very small gradients in travelling block, this case frequently happened because very slow drilling. The main condition of "Circulation" state is that travelling block should be static i.e. the block position sensor data does not change, the main reason of confusing this state with other states is the gradient of block position sensor data. Practically, it will be very difficult to have "Circulation" state without any movement in travelling block, most of the times, the block position fluctuated for few centimeters and sometimes more and then the circulation is detected as another state due to states of other sensor data.

Test Data Set: TD1203

Total Accuracy: 96.59%

		Detected Rig States							
		Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Actual Rig States	Drilling ROT	**536**	23	0	5	0	16	0	**92.41%**
	Run in Hole	12	**1765**	0	13	5	21	0	**97.19%**
	Pull out of Hole	0	0	**1534**	0	23	13	0	**97.71%**
	Reaming	4	23	0	**978**	0	10	3	**96.07%**
	Back Reaming	0	0	13	0	**858**	21	2	**95.97%**
	Circulation	5	10	12	9	32	**1376**	5	**94.96%**
	MakeCon	1	3	6	4	0	0	**1283**	**98.92%**
	Recall	**96.06%**	**96.77%**	**98.02%**	**96.93%**	**93.46%**	**94.44%**	**99.23%**	**96.59%**

Test Data Set: TD969

Total Accuracy: 97.36%

		Detected Rig States							
		Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Actual Rig States	Drilling ROT	**89**	2	0	4	0	7	6	**82.41%**
	Run in Hole	**1**	**1984**	0	13	0	12	0	**98.71%**
	Pull out of Hole	**0**	0	**2236**	0	21	10	0	**98.63%**
	Reaming	**5**	33	12	**137**	14	0	0	**68.16%**
	Back Reaming	**0**	0	14	**0**	**148**	9	15	**79.57%**
	Circulation	**3**	11	0	21	0	**291**	4	**88.18%**
	MakeCon	**0**	0	5	12	4	0	**3890**	**99.46%**
	Recall	**90.82%**	**97.73%**	**98.63%**	**73.26%**	**79.14%**	**88.45%**	**99.36%**	**97.36%**

Test Data Set: TD987

Total Accuracy: 98.60%

		Detected Rig States							
		Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Actual Rig States	Drilling ROT	**241**	10	0	0	0	5	4	**92.69%**
	Run in Hole	**16**	**3096**	0	27	0	0	0	**98.63%**
	Pull out of Hole	**0**	0	**2982**	14	12	0	4	**99.00%**
	Reaming	**12**	0	12	**1959**	0	0	6	**98.49%**
	Back Reaming	**0**	43	0	**0**	**1935**	13	0	**97.19%**
	Circulation	**0**	0	5	3	0	**2234**	0	**99.64%**
	MakeCon	**0**	3	0	0	10	0	**1576**	**99.18%**
	Recall	**89.59%**	**98.22%**	**99.43%**	**97.80%**	**98.88%**	**99.20%**	**99.12%**	**98.60%**

FIGURE 9.7: Confusion Matrices of TD1203 - TD969 - TD987

9.6.3 Test Rigs: TD56, TD287 and TD285

Figure 9.8 shed the light on more three data sets **TD56, TD287** and **TD285** with accuracy between 95% and 98%. The results confirm the previous conclusions on "Circulation" and "Make Connection" states. The states of "Back Reaming" is confusion in few cases with "Drilling ROT" and this is due to the wrong segmentation of block position in PLA Algorithm, this is heavily related to the error of block position parameter, because this parameter controls the segmentation process and how much the changes will be detected at block position sensor data, the smaller is the error, the more fine segments will be detected. The reason of confusing "Make Connection" state with "Run in Hole" is the parameter of Expectation Maximization algorithm which is used to detect the hookload threshold that separates "InSlips" state from "OutOfSlips".

Test Data Set: TD56

Total Accuracy: 94.99%

		Detected Rig States							
		Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Actual Rig States	Drilling ROT	**79**	3	0	14	0	0	0	**82.29%**
	Run in Hole	**8**	**723**	0	9	0	2	0	**97.44%**
	Pull out of Hole	**0**	0	**616**	**0**	13	1	0	**97.78%**
	Reaming	**3**	17	**0**	**270**	0	0	1	**92.78%**
	Back Reaming	**0**	0	12	**0**	**214**	15	2	**88.07%**
	Circulation	**0**	6	4	2	0	**180**	0	**93.75%**
	MakeCon	**0**	0	1	3	2	0	**155**	**96.27%**
	Recall	**87.78%**	**96.53%**	**97.31%**	**90.60%**	**93.45%**	**90.91%**	**98.10%**	**94.99%**

Test Data Set: TD287

Total Accuracy: 98.00%

		Detected Rig States							
		Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Actual Rig States	Drilling ROT	**241**	17	0	4	0	0	0	**91.98%**
	Run in Hole	11	**1689**	0	8	0	0	0	**98.89%**
	Pull out of Hole	0	0	**1737**	0	24	5	3	**98.19%**
	Reaming	1	0	**0**	**1959**	**0**	7	0	**99.59%**
	Back Reaming	0	0	26	**0**	**1935**	25	0	**97.43%**
	Circulation	0	0	31	13	21	**1834**	13	**95.92%**
	MakeCon	**0**	0	5	7	3	0	**1576**	**99.06%**
	Recall	**95.26%**	**99.00%**	**96.55%**	**98.39%**	**97.58%**	**98.02%**	**98.99%**	**98.00%**

Test Data Set: TD285

Total Accuracy: 95.89%

		Detected Rig States							
		Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Actual Rig States	Drilling ROT	**154**	6	0	0	0	2	0	**95.06%**
	Run in Hole	4	**742**	0	21	0	0	0	**96.74%**
	Pull out of Hole	**0**	0	**842**	**0**	9	6	0	**98.25%**
	Reaming	**2**	3	9	**576**	**0**	8	3	**95.84%**
	Back Reaming	**4**	9	17	**0**	**471**	0	10	**92.17%**
	Circulation	**1**	7	0	0	8	**272**	0	**94.44%**
	MakeCon	**0**	24	3	0	0	0	**587**	**95.60%**
	Recall	**93.33%**	**93.81%**	**96.67%**	**96.48%**	**96.52%**	**94.44%**	**97.83%**	**95.89%**

FIGURE 9.8: Confusion Matrices of TD56 - TD287 - TD285

9.6.4 Test Rigs: TD101 and TD5

Figure 9.9 shows the data sets **TD101** and **TD5** with accuracy value around 97%. In confusion matrix of **TD5**, it is clear that the system confuses "Drilling ROT" with other states, this is due to the bad quality of hole depth and bit depth sensor data, the data quality issue in this case is that hole depth and bit depth are close to each other and bit depth is not changes during "Reaming", "Running in Hole", or "Back Reaming".

Test Data Set: TD101

Total Accuracy: 96.68%

Detected Rig States

Actual Rig States	Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Drilling ROT	**204**	35	0	27	0	8	0	**74.45%**
Run in Hole	9	**3891**	0	19	0	24	21	**98.16%**
Pull out of Hole	0	5	**3287**	7	17	19	23	**97.89%**
Reaming	4	28	8	**1492**	0	8	6	**96.51%**
Back Reaming	0	17	57	4	**1520**	3	7	**94.53%**
Circulation	0	24	49	36	16	**1612**	31	**91.18%**
MakeCon	0	0	5	0	0	24	**3756**	**99.23%**
Recall	**94.01%**	**97.28%**	**96.51%**	**94.13%**	**97.88%**	**94.94%**	**97.71%**	**96.68%**

Test Data Set: TD5

Total Accuracy: 96.83%

Detected Rig States

Actual Rig States	Drilling ROT	Run in Hole	Pull out of Hole	Reaming	Back Reaming	Circulation	Make Connection	Precision
Drilling ROT	**364**	10	0	5	0	6	0	**94.55%**
Run in Hole	32	**2381**	0	14	0	13	0	**97.58%**
Pull out of Hole	21	0	**2256**	**0**	23	2	14	**97.41%**
Reaming	16	14	0	**1383**	1	16	3	**96.51%**
Back Reaming	0	0	21	**0**	**1219**	15	0	**97.13%**
Circulation	18	26	6	10	4	**1600**	0	**96.15%**
MakeCon	0	24	18	0	21	0	**1574**	**96.15%**
Recall	**80.71%**	**96.99%**	**98.04%**	**97.95%**	**96.14%**	**96.85%**	**98.93%**	**96.83%**

FIGURE 9.9: Confusion Matrices of TD101 - TD5

9.6.5 Results Benchmarking

Confusion matrix is good in getting a comprehensive idea on the detection and recognition results, but still missing is how the results look like from statistical viewpoint. Most of the decisions on rig states are taken based on the statistical distribution of states values. For example, the statistical distribution of Make Connection states gives an idea on the performance of the rig and drilling crew, also this will help in comparing the performance of different crews and rigs. Comparing the rig states detection and recognition results to a pre-classified rig states will give a deep insight on the results and help more in finding out the differences between the results and analysing the reasons behind those differences. In this section, a comparison between rigs states obtained from TDE proNova systems for rig states detections and the suggested rig states detection in this

thesis. It is important to mention here that the states, which are obtained from proNova systems, are refined and corrected manually by drilling experts. The term "Old" refers to the results from proNova systems, the term "New" refers to the results produced by new suggested system. The data set contains sensors data from 2000m well drilled in 42.5 days, the data is received at 0.2 Hz frequency.

Figure 9.10 shows a comparison between "Old" and "New" results for rig states: Drill, Reaming, BackReaming, and Moving Out. The results are shown as histogram of each rig state with a mean value plotted as red line on the histogram.

Drill State shows a perfect match between the old results from experts (90) and the new suggested system (89), the mean values of drill state operations (54.17 mins from Old and 54.43 from New) and the histogram of the states give almost a full match.

Reaming State refers to a different between the two systems, the histogram of Old system is positive skewed and this returns to the way that the old system applied to detect Reaming state, the old system used slope and direction of block position to determine the state of reaming and this makes it very sensitive to all the small changes happened to block position i.e. if the driller lifted the drill-string up sometime may it happens that the drill-string is fluctuated during the reaming operation and this fluctuation detected as many reaming and back reaming states in the old system. The PLA in suggested system is more stable to the fluctuations of drill-string.

BackReaming State has the same discussion of **Reaming State** and this explains why the histogram of BackReaming is positively skewed and the mean value of BackReaming in new system (1.94 mins) is more than mean value of old system (0.93 mins).

MovingOut State shed the light on interesting difference between the old system and new system, the big difference in samples number, where it is in the old system 4157 MovingUp states in the old system and 2475 MovingUp states in new system, The reason behind this difference is the sensitivity of old system to the block position movements, the old system can detect two or three adjacent MovingUp states while the new system detects one MovingUp state, the stability of new systems comes from the stability of Piecewise Linear Approximation algorithm used to detect the trends on Block Position sensor data. Figure 9.11 shows a comparison of rig states: MovingIn, MakeConnection and Circulation.

MovingIn State has good match between old and new results.

Make-Connection State also refers to a perfect match between the old and new results, the difference in mean values between the old and the new results returns to the difference in boundaries of detected InSlips states, the old system depends only on a threshold to detect the boundaries of InSlips/MakeConnection and it performs no adjustment to the boundaries of detected InSlips/MakeConnection state, the new system does a boundaries adjustments as shown in chapter 7.

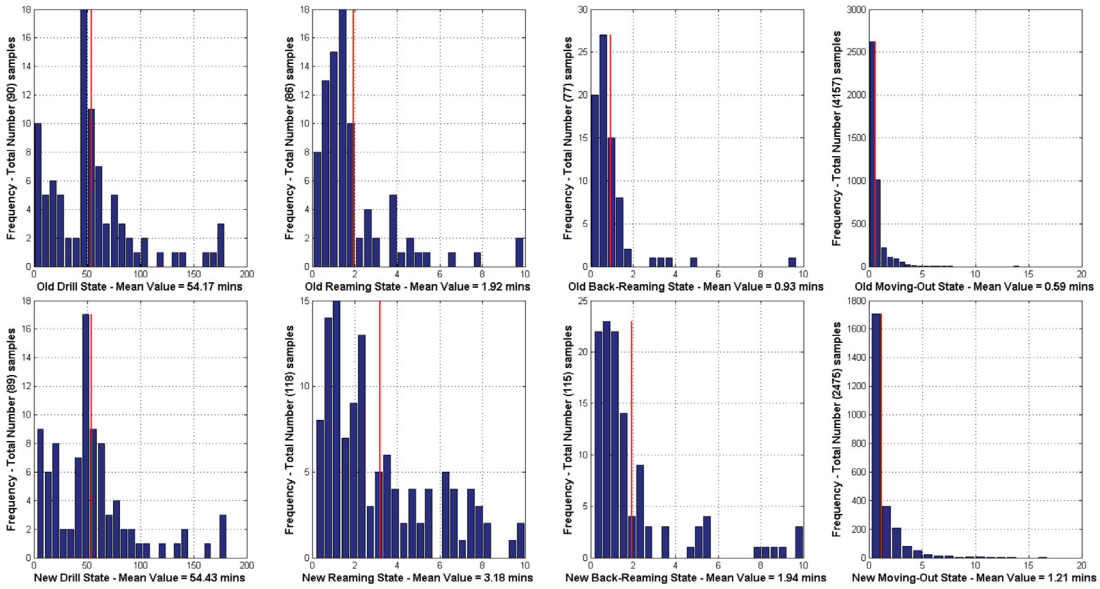


FIGURE 9.10: Comparing results of Drill - Reaming - BackReaming - MovingOut states.

Circulation State has a big difference in samples numbers between old and new results and this is because of fluctuations problem in detecting block position slope based on two adjacent values in old system and using trend approximation techniques in new system.

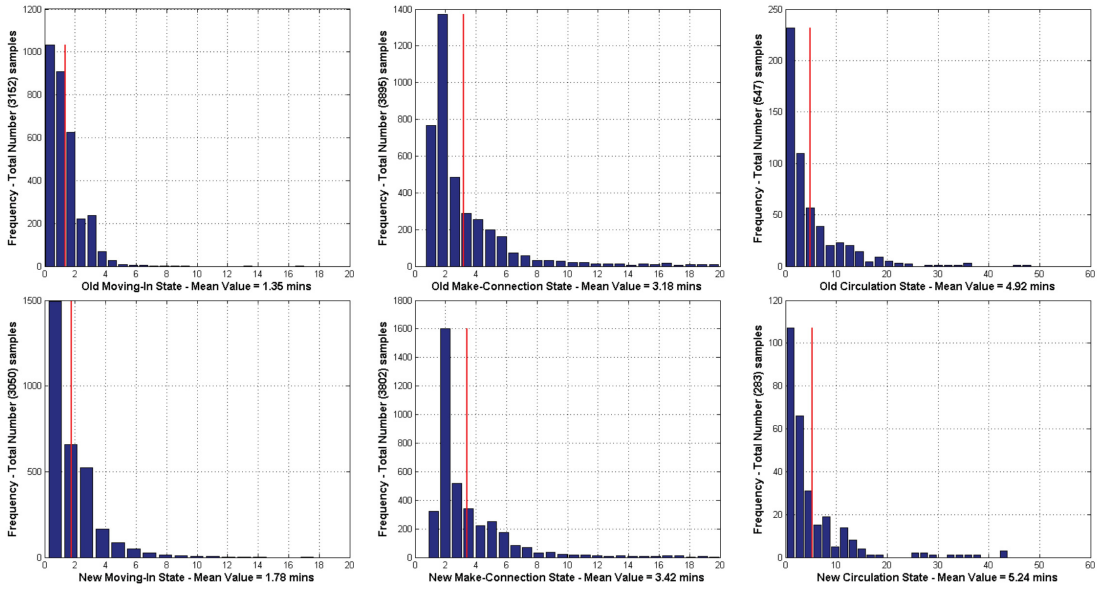


FIGURE 9.11: Comparing results of Moving In - Make Connection - Circulation states.

9.7 Performance Evaluation

In this paragraph, a brief on algorithms complexity will be discussed then evaluation of the system performance on test data sets will be presented. The machine which is used to do the performance tests is Windows(R) machine with 8 GB RAM and Intel(R) Dual Core(TM) i5 CPU @2.40 GHz.

9.7.1 Theoretical Run-time Complexity

The sequence of execution is summarized by the execution flow at figure 9.1, the complexity of the system will be the highest-order of all complexities of the algorithms.

Hookload Segmentation Complexity The execution flow starts with Expectation Maximization algorithm or Envelope Algorithm to determine the threshold and then it uses this threshold to segment the hookload. Expectation Maximization algorithm (see chapter 6) iterates over all the elements of data sets n by the number of clusters k, this process repeated with the difference of log-likelihood is greater than stop error, the number of iterations can be α, then the complexity will be $O(n * k * \alpha)$. The envelope algorithm starts with a loop of a window of size ws, then the envelope is calculated based on this window, then the complexity will be: $O(n/ws)$. The final step is the complexity of hookload segmentation algorithm using the calculated threshold, this algorithm is a simple loop to evaluate each data points if it is above or under the threshold value, $O(n)$.

Block Position Segmentation: Piecewise Linear Approximation algorithm complexity can be estimated using the discussion in [82]. The PLA complexity for one "OutOfSlips" segment is given as $O(c * L)$ where L is the length of "OutOfSlips" segment because PLA is applied on block positions just for "OutOfSlips" segments, and c is the length of block position segments during "OutOfSlips". The number of "OutOfSlips" segments is $NS = L/n$. Then the complexity of Block Position Segmentation will be $O(NS * L * c)$.

RPM Segmentation: If the system configured to do the rpm segmentation using manually-specified threshold then the complexity will be $O(n)$, but if the configuration of using Envelope algorithm the complexity willl be $O(n/ws)$ where ws is the window size.

Flowrate Segmentation: If the system configured to do the flowrate segmentation using manually-specified threshold then the complexity will be $O(n)$, but if the configuration of using Envelope algorithm the complexity willl be $O(n/ws)$ where ws is the window size.

Depth Segmentation: The complexity of segmenting the sensor data based on depth is given by $O(n)$, because the threshold of "OnBottom" state configured manually by experts and one iteration over all the data is enough to be segmented.

Segments Recognition: In this phase all the obtained segments will be processed and their properties will be evaluated to which rig state each segment should be assigned, this can be performed by one iteration over all the segments then the complexity will be $O(S)$ where S is the number of obtained segments and $S \ll n$

Complexity of State Detection System: Using Expectation Maximization for hookload segmentation, the complexity of detection system will be:

$$HighestOrder(O(n*k*\alpha), O(NS*L*c), O(n/ws), O(n/ws), O(n), O(S)) \approx O(n*NS). \tag{9.1}$$

Using Envelope algorithm for hookload segmentation, the complexity of detection system will be:

$$HighestOrder(O(n/ws), O(NS*L*c), O(n/ws), O(n/ws), O(n), O(S)) \approx O(n*NS). \tag{9.2}$$

The best case of this complexity when there is just one segment $NS = 1$, then the complexity will be close to $O(n)$. And the worst case is when there is a segment between each two data points i.e. $NS = n/2$, and this means the complexity, in this case, will approach to be $O(n^2)$.

9.7.2 Performance Tests

It is obvious that the run-time is not linear to the number of data points and this confirms the theoretical estimation of the complexity. The tests shows that the more the data points, the complexity getting closer to be quadratic to n. Figures 9.12-9.12-9.13-9.14-9.15-9.16-9.17 display the results of performance tests on each data set, and all of them prove that the run-time fits with the theoretical study of the system's complexity.

Figure 9.15 shows three data sets collected from different rigs. It is clear that the run-time is different between those data sets. This result comes from the fact that each of those rigs has different length of drill-stand, the rigs of **TD1203** and **TD1258** have a short drill-stand with length $\approx$ 15 meters, and the third rig **TD1246** has a long drill-stand of 30 meters. Furthermore, the data sets of the short stand rigs are processed with PLA Error parameter 0.1 while the data set from the third rig is processed with PLA

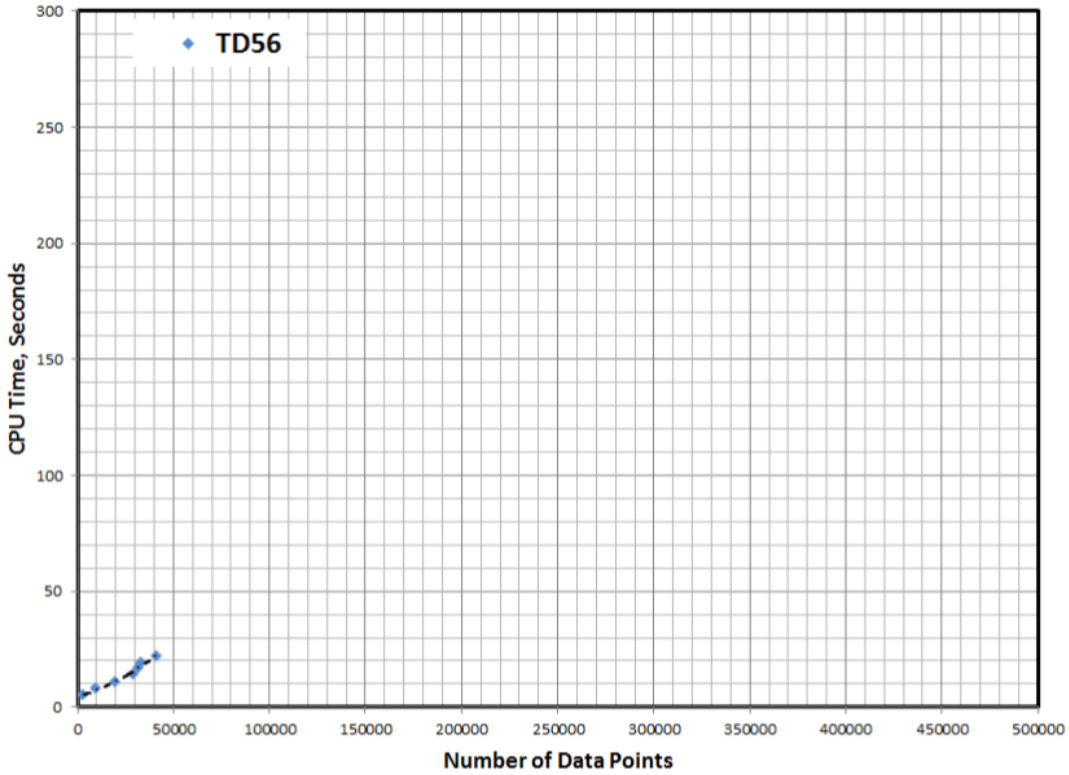


FIGURE 9.12: Rig state detection performance for test data set TD56.

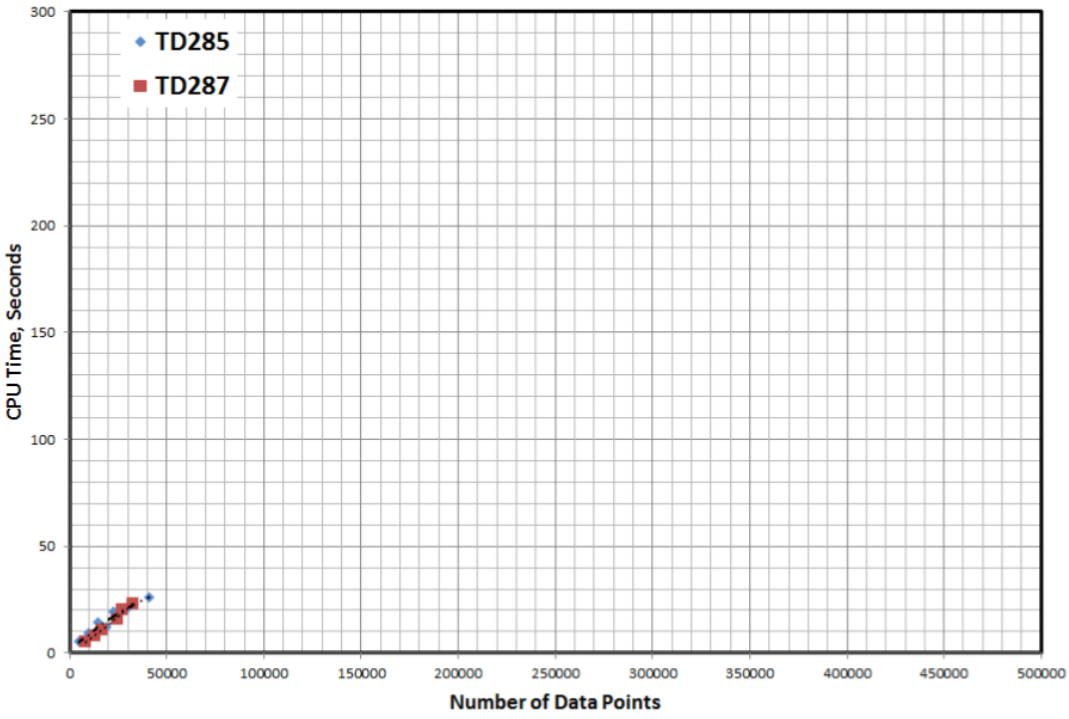


FIGURE 9.13: Rig state detection performance for test data sets TD285 and TD287.

Error parameter 10, then the PLA error and length of drill-stand has the main impact

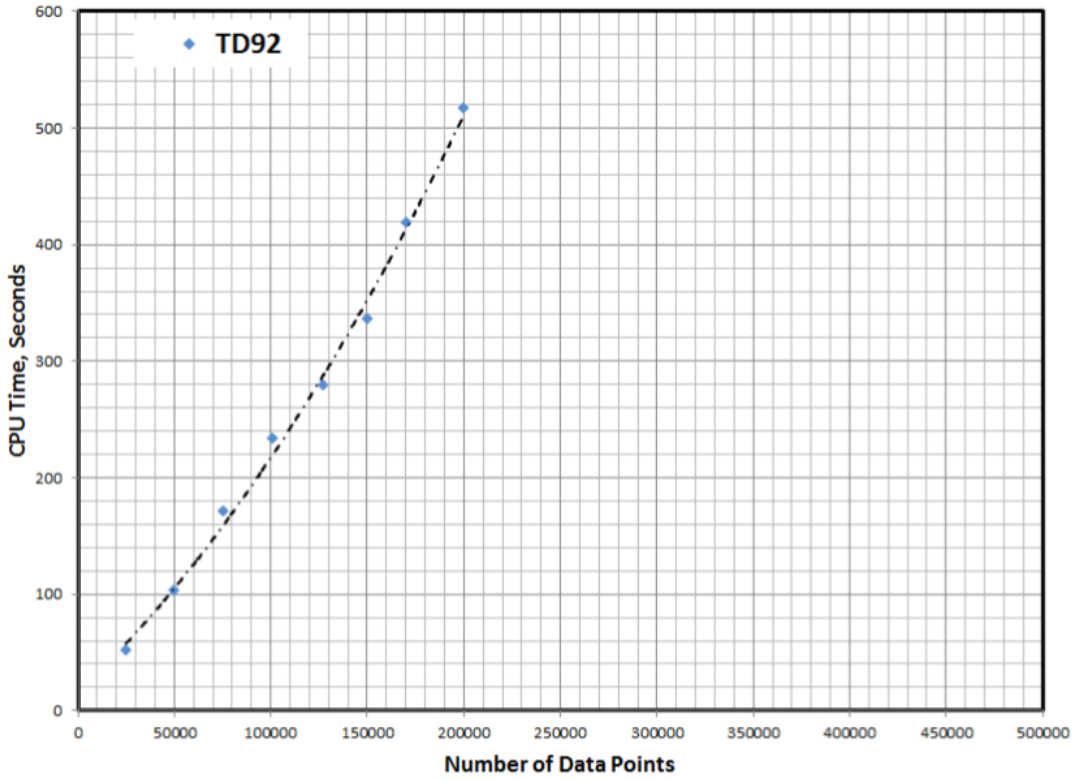


FIGURE 9.14: Rig state detection performance for test data set TD92.

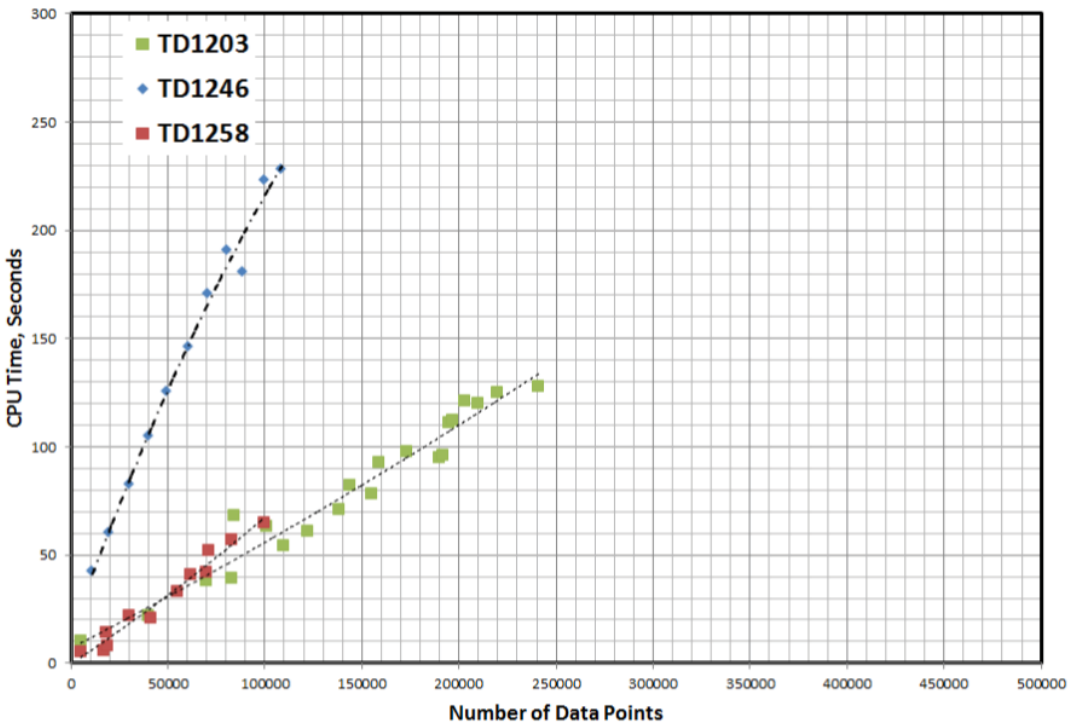


FIGURE 9.15: Rig state detection performance for test data sets TD1203, TD1246, and TD1258.

on the performance.

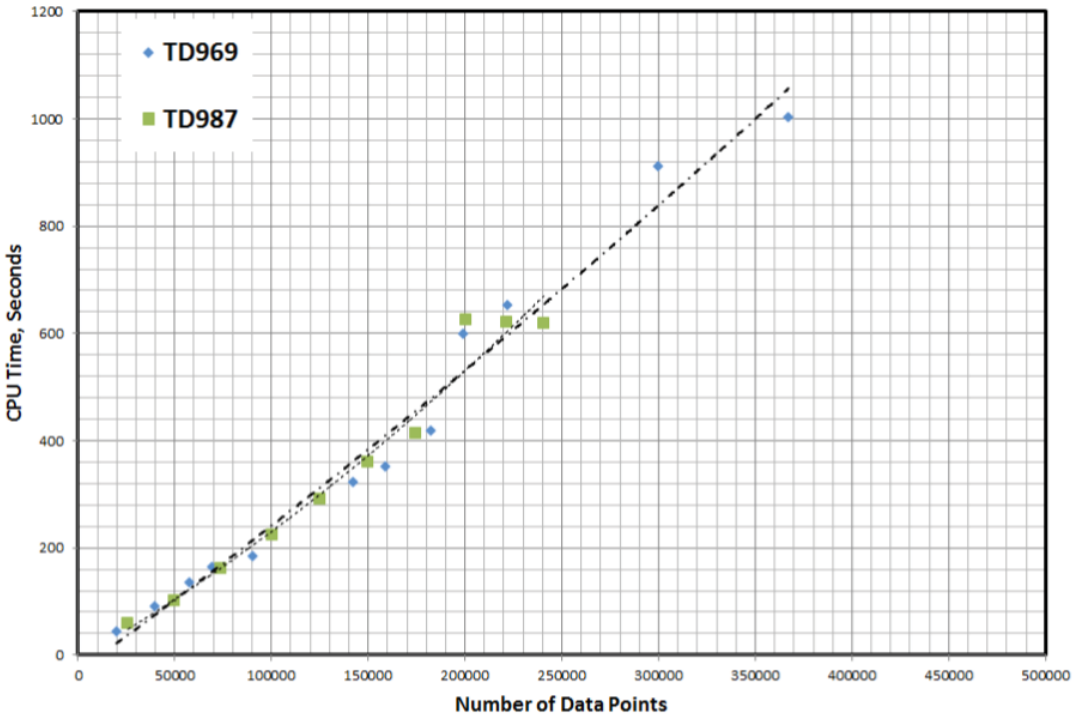


FIGURE 9.16: Rig state detection performance for test data sets TD969, and TD987.

Figure 9.18 shows the detailed run-time, and it separates the run-time required by PLA. It is clear that PLA has huge impact on the overall performance, it takes around more than 75% as average of the total run-time.

9.8 Summary

This chapter demonstrated detection system of rig states from sensor data, it showed how the sensor data was processed to conclude the segments that represent states of the rig. The basic concept behind the detection system is the rig state knowledge that links rig states with sensors states. Distributed architecture was suggested to implement the detection system, then a prototype of the system was demonstrated and used on eleven sensor data sets from different rigs. The results showed that the accuracy of detection system and where the confusions are and the reasons behind them. Complexity study and Performance evaluation were performed on the test data sets, the results showed that the run-time is linear with size of processed data and this matches the theoretical complexity of the system. Piecewise Linear Approximation algorithm is a resource consuming algorithm and it shows high sensitivity for gradient of block position which

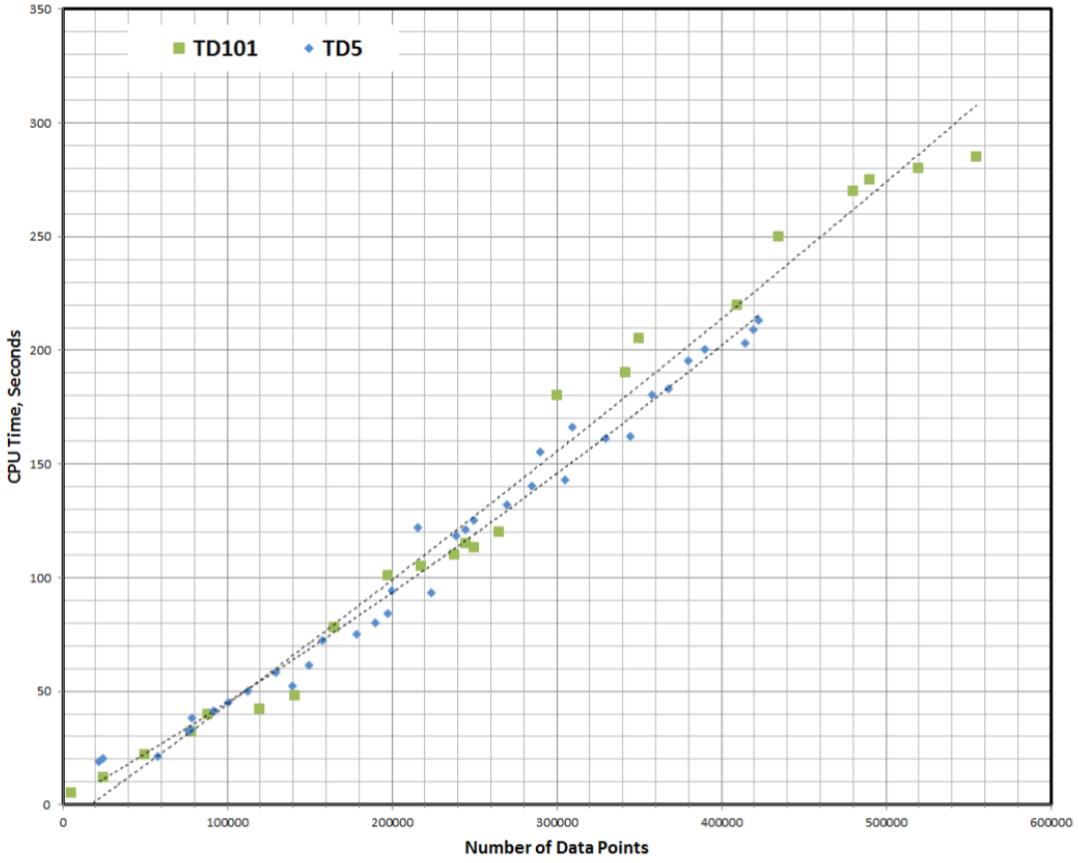


FIGURE 9.17: Rig state detection performance for test data sets TD101, and TD5.

is considered as a source of confusion in detection many states. PLA and block position segmentation has a big potential for improvements in the future.

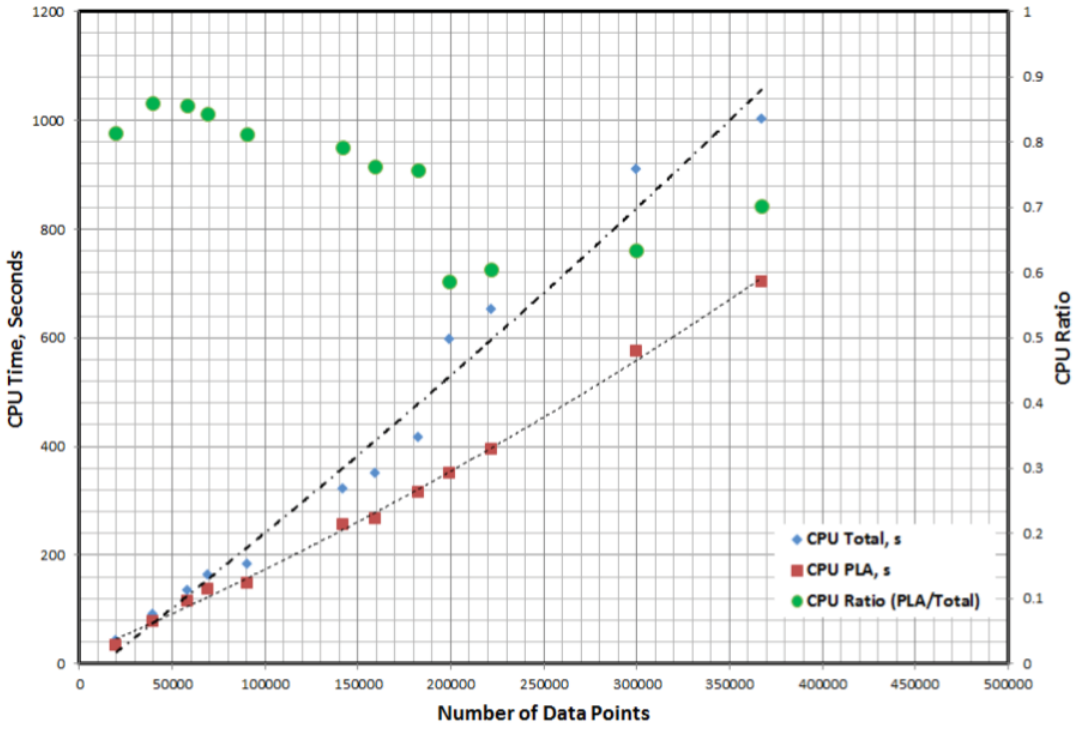


FIGURE 9.18: PLA algorithm performance test over TD969.

Part III

Conclusion and Future Work

Chapter 10

Summary, Conclusion and Future Work

10.1 Summary

A rig state detection system from sensor data has been developed, which features the following characteristics:

- A distributed bridge component based on WITSML standard to fetch sensors data from rig site has been developed and tested on real rigs (Appendix A).
- The system fuses the information from different rig sensors data to recognize the rig state. The fusion process was designed based on Waterfall and Distributed Blackboard Multisensor Fusion models.
- The system integrates prior knowledge of drilling processes with suggested steps in fusing and concluding the information from sensor data.
- Three thresholding algorithms were tested and applied on real and simulated rig sensor data to detect InSlips/OutOfSlips states from Hookload sensor data. Expectation Maximization , Otsu thresholding, and Envelope algorithms were tested, the results showed an advantage to use Envelope algorithm over other algorithms due to its high accuracy and flexibility to quality problems in hookload sensor data.
- The system is able to detect all movements of drill-string via monitoring the trends and changes on Block Position sensor data. Piecewise Linear Approximation algorithm is used by the system to detect and approximate the trends in Block Position sensor data, PLA shows tolerance to noisy data.

- Correcting to boundaries of InSlips/OutOfSlips states is performed by the detection system to match the detected states with the engineering definition of InSlips/OutOfSlips concept. Furthermore, a validation of the pattern shape of Hookload and Block Position sensors data is applied to differentiate between InSlips and MakeDis/Connection states.
- A distributed architecture using Middelware is used to implement the system with different segmentation software components to fetch the data from rigsite and then detect the states of the rig.
- The system is applied on sensor data of 11 data sets (330 drilling days) collected from different rigs distributed over the world and from different types of drilling rigs: drilling ships, land drilling rigs, drilling jack ups and drilling platforms.
- The detection and recognition results are compared to results classified by TDE proNova System with manual correction from drilling experts. No manual interaction or modification is applied on the results of suggested system. The accuracy results shows a value range between 92% and 99%.
- A statistical comparison is performed on one data set and it shows a high percentage of statistical matching between old and new results.
- Main restriction found in old proNova technology is its sensitivity to the fluctuation of block position sensor data where this problem is solved by applying PLA in the new suggested system.
- Another restriction found in old proNova technology is the usage of one fixed threshold while the new system uses the concept of adaptive threshold in detecting different states of rig's machines from sensor data.
- Adaptive thresholding reduces the configuration process complexity and maximizes usage of the system with minimum interference from the engineers.
- Due to performance analysis of the system, it is shown that the overall performance of the system is linear, this result was concluded theoretically and proofed practically on all test data sets.

10.2 Conclusion

The following main conclusions can be drawn from the analysis of rig sensor data in purpose of detecting different rig states:

- Rig sensors surface measurements can be used to detect all drilling rig's activities that the rig's machines are used in.
- Using of thresholding concept plays a main role in finding the start and end of each detected InSlips/OutOfSlips state.
- The concept of trends detection using piecewise linear approximation helps finding out the start and end of each detected drill-string movements with high accuracy.
- Improving InSlips/OutOfSlips detection using boundaries adjustment algorithms raises the accuracy of start/end states determination.
- Using distributed WITSML bridge component based on Web Services helps in getting the rig sensors data from remote sites located somewhere in the world.
- Each sensor measurements can be used to monitor state of different rig's machine, for example: Hookload sensor data is used to monitor state of drill-string situation (InSlips/OutOfSlips), Block Position sensor data used to monitor the movements of drill-string (Up/Down/Static), RPM refers to the state or rotary system (YES/NO), flowin rate gives indication to the state of mud pumps (ON/OFF), bit and hole depth estimates whether the drill-bit touches the bottom of hole or it is far away from hole bottom.
- The uncertainty of state detection is estimated when the segmentation is applied: InSlips/OutOfSlips thresholding uncertainty is estimated by measuring the distance of threshold from the centers of data clusters, while the uncertainty of drill-string movements segmentation is evaluated by estimating the certainty at each joint of two adjacent segments. The overall uncertainty of detection process is evaluated through the confusion matrix or statistical benchmarking with pre-classified sensor data sets.
- The detection process can be run at any time instance and no need to apply it at the beginning of drilling work.
- Detection of InSlips/OutOfSlips requires one parameter in case of using Envelope Algorithm, the parameter is BHA weight which is the minimum weight to stop detection between two drilling phases. In case of using EM algorithm to detect InSlips/OutOfSlips, the clusters number is required to make the algorithm run. Block Position segmentation process requires an error parameter to determine the granularity degree of segments to be detected. OnBottom tolerance parameter is used to determine whether the drill-bit is on hole bottom or not.

- The advantage of using distributed architecture is that components of the system are loosely-coupled which means that if one of the components failed in doing its calculation then other components will not be effected.
- The middleware provides the infrastructure for implementing distributed applications of rig state detection with minimum effort, the development of each segmentation component was done independently from the development of the others.

10.3 Future Work

In the author's opinion, there are a number of open issues directly related with the work presented in this thesis:

- PLA algorithm consumes around 75% of performance as it is shown in figure 9.18 and it can be improved by applying different approach rather than bottom-up approach. Keogh et al. suggested to use SWAB algorithm instead of using bottom-up due to its high performance [82].
- The parameter of Block Position error can be revised by doing an in depth study on the parameter values and the detected segments on block positions.
- Better estimation to the state of OnBottom based on re-calculation of bit depth and hole depth based on InSlips/OutOfSlips states and block position, this will remove the usage of OnBottom tolerance parameter and makes better estimation of OnBottom state.
- The detection system can be extended to detect all kind of rig's states based on different and other sensors data such as detecting the states of testing Blowout Preventer BOP using detecting of different states of pumps's pressure.
- The system can be extended to detect Drilling ROCKING mode by analyzing RPM data to detect the fluctuated behaviour of Rotary System during Drilling ROCKING.
- Detecting higher-level of operations such as Drilling Runs, Jobs and Phases can be performed by introducing more sensors data.
- The results of rig state detection can be used to validate the information in activities section in the daily morning reports of the rig.

Appendix A

Appendix A: Distributed Sensor Data Acquisition

A.1 Introduction

This appendix describes the sensor data formatting and how this data will be transferred using distributed service-oriented architecture from rig-site to office site.

Usually drilling rigs are located far away from office location and there is an urgent need to obtain sensor data available at office site in a reasonable amount of time. Industrial oil companies started the initiative of WITS which is a specification for wellsite information transfer to remote locations [91]. The idea has been developed further and WITSML is suggested as a standard to format wellsite - rigsite - information using XML language, and then using web-services to transfer the data from rig site to office site over satellite Internet connection [40].

A.2 Wellsite Information Transfer Specification

The WELLSITE INFORMATION TRANSFER SPECIFICATION (WITS) is a communications format used for the transfer of a wide variety of wellsite data from one computer system to another. At the time when WITS was proposed, the technology did not help to transfer the data from rig site to remote locations, so the WITS was used just to encode the data and store it at local storage devices to be transfer physically to office site.

WITS is a multi-level format which offers an easily achieved entry point with increasingly flexible higher levels. At the lower levels, a fixed format ASCII data stream is employed,

while, at the highest level, a self-defining customizable data stream is available. A WITS data stream consists of discrete data records. Each data record type is generated independently of other data record types and each has a unique trigger variable and sampling interval. WITS also defines a basic set of pre-defined records to which user-defined record types may be added (see figure A.1).

A.2.1 WITS Level 0

Also known as "Intra Rig Transfer Specification", this involves a very basic ASCII transfer format intended primarily for sharing of information between service companies. Data items are identified by a numeric string tying the value to a particular location within a Pre-Defined Record, or to an agreed upon addition to the Data Dictionary [40].

A.2.2 WITS Level 1

At level 1 and above, the data stream takes on a binary (LIS - Log Information Standard) format. Values are expressed in LIS-defined representations (e.g. floating point, integer, string, etc). The data items are packaged into a WITS Data Record and then sandwiched between LIS Physical and Logical Record Headers and Trailers, to make up a LIS Data Record. Twenty five Pre-Defined Records have been identified, covering, among other areas, drilling, geology, directional work, MWD, cementing and testing. At Level 1, these data records, generated at varying times and under varying rig conditions, are constructed and placed in the data communications channel. No LIS record types besides Data Records are used at this level. Each of the 25 Pre-Defined Records has a fixed size in bytes. However, each contains designated 'spare' channels for limited customization. The Pre-Defined Records are showed in figure A.1 [40].

A.2.3 WITS Level 2

WITS Level 2 builds on Level 1 through addition of WITS bidirectional dialogue through the use of LIS Comment records. This dialogue is used in synchronization at start up and after a communications line interruption, as well as permitting two-way messaging between the sender and receiver. Such messages might include requests for change in transmission intervals for certain records [40].

Rec	Name	Description
1	General Time-Based	Drilling data gathered at regular time intervals
2	Drilling - Depth Based	Drilling data gathered at regular depth intervals
3	Drilling - Connections	Data gathered at drilling connections
4	Hydraulics	Hydraulics data gathered while circulating
5	Trip - Time	Tripping data gathered while running in/pulling out
6	Trip - Connections	Tripping data gathered at tripping connections
7	Survey/Directional	Directional/Survey data
8	MWD Formation Evaluation	MWD Formation Evaluation data
9	MWD Mechanical	MWD Mechanical data
10	Pressure Evaluation	Pressure Evaluation data
11	Mud Tank Volumes	Mud Tank (Pit) Volume data
12	Chromatograph Cycle-Based	Chromatograph Cycle data
13	Chromatograph Depth-Based	Chromatograph data averaged over depth intervals
14	Lagged Mud Properties	Mud Property data based returns depth increments
15	Cuttings / Lithology	Cuttings Lithology and related data
16	Hydrocarbon Show	Hydrocarbon Show related data
17	Cementing	Well Cementing operations data
18	Drill Stem Testing	Well Testing operations data
19	Configuration	Drillstring and Rig Configuration data
20	Mud Report	Mud Report data
21	Bit Report	Bit Report data
22	Comments	Freeform Comments
23	Well Identification	Well Identification data
24	Vessel Motion / Mooring Status	Vessel Motion and Mooring Status data
25	Weather / Sea State	Weather and Sea State data

FIGURE A.1: WITS Pre-Defined Record Types

A.2.4 WITS Level 2b

WITS Level 2b adds the option to buffer data that has been transmitted, making it available for re-transmission in the event of non-receipt of data by the receiver [40].

A.2.5 WITS Level 4

WITS Level 4 employs a completely different format than the previous levels since it is based on the emerging data transfer standard of API RP66. The concepts of Pre-Defined Records and Bi-Directional Dialogue remain, but using RP66 as the formatting mechanism [40].

A.2.6 Limitations of WITS

The recognized limitations of the current WITS specification are:

- Outdated MWD data records.
- Data is record driven, not object oriented.
- Restrictions on number of drill string and casing sections.
- Inflexibility for handling data in different units of measurement.
- Limited capacity for handling static well information.
- WITS records are extensible but not self-describing.
- Use of binary data formats is not platform independent.

A.3 Wellsite Information Transfer Standard Markup Language - WITSML

The Wellsite Information Transfer Standard Markup Language (WITSML) consists of XML data-object definitions and a web services specification developed to promote the right-time, seamless flow of well data between operators and service companies, as well as regulatory agencies, to speed up and enhance decision-making and reporting [92].

The creation of WITS record involves a mapping process from individual tool sensor output values to a sequential numerical data stream [93], sometimes the crew changing the units of measurements or the order of the channels maybe changed for any reason, this causes many problems at the receiver side, and even some of those problems can not immediately discovered and this may takes one complete day at the office side to figure out the source of problem. This is considered as a big need for WITSML where the meta data will be sent always with the raw data, so any changes of the order of the sensors or the units, will then be transferred directly with the data. Figure A.2 shows an example on WITS records and how they are represented using WITSML [93].

A.4 Distributed Sensor Data Acquisition Unit - WITSML Bridge

WITSML has a standard for data description using of WITS + XML and a standard for data transfer using distributed web-service enabled software systems [41]. Usually at the rig site, the sensor data is acquired using mudlogging systems of service companies, then this data will be transferred using WITS format over serial port to the WITSML

WITS Level 0	WITSML
&& 0801TEST-Well -559 080816001.77 08150.138 082320.73 !! && 0801TEST-Well -559 080816002.77 08150.152 082326.52 !! && 0801TEST-Well -559 080816003.77 08150.165 082330.12 !!	<logCurveInfo uid="TEST-Well-559"> <mnemonic>DEPTMEAS</mnemonic> <unit>ft</unit> <mnemAlias>md</mnemAlias> <minIndex uom="ft">16001.77</minIndex> <maxIndex uom="ft">16003.77</maxIndex> <columnIndex>2</columnIndex> <curveDescription>Total Depth</curveDescription> <typeLogData>double</typeLogData> <mnemonic>MR1</mnemonic> <unit>OHMM</unit> <mnemAlias>MR1</mnemAlias> <minIndex uom="OHMM">0.138</minIndex> <maxIndex uom="OHMM">0.165</maxIndex> <columnIndex>2</columnIndex> <curveDescription>Resis 1 reading </curveDescription> <typeLogData>double</typeLogData> <mnemonic>MG1</mnemonic> <unit>API</unit> <mnemAlias>MG1</mnemAlias> <minIndex uom="API">20.73</minIndex> <maxIndex uom="API">30.12</maxIndex> <columnIndex>3</columnIndex> <curveDescription>Gamma Ray 1 reading </curveDescription> <typeLogData>double</typeLogData> </logCurveInfo> <data>16001.77, 0.138, 20.73</data> <data>16002.77, 0.152, 26.52</data> <data>16003.77, 0.165, 30.12</data>

FIGURE A.2: Sensor data represented using WITS and WITSML [93]

server, at WTSIML server the data is stored locally and it will be ready to be sent when any request from other parties is received.

The WITSML server follows the APIs of WITSML standards. The WITSML server can follow one of two data exchange paradigms: Subscribe/Publish through Publish Interface or Request/Response through Store Interface [40]. Using Subscribe/Publish paradigm, the WITSML server receives subscriptions of data objects that the clients are interested in. Once the data on subscribed objects is available, the WITSML server publishes the data back to the client. The other Request/Response paradigm is that the client sends a data query or request to the server and then the server will directly send the response with data back to the client.

Figure A.3 shows the architecture of WITSML Bridge which is software component responsible of fetching sensor data from rig site. WITSML Bridge is used to query WITSML server for the sensors data available at the server and then it stores the data locally at office site or publishes the data to other interested parties.

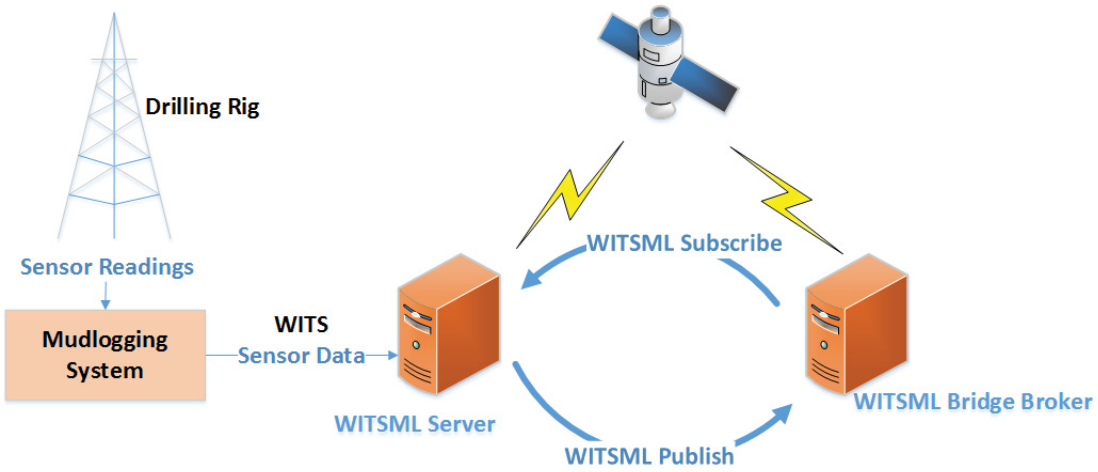


FIGURE A.3: Distributed Data Acquisition using WITSML Bridge with Publish/Subscribe Model

The WITSML server plays the role of middleware between the measurements systems (mudlogging systems) of service companies and other parties. The big advantage here is that the sensor data or any wellsite information will be available to any interested party once this data is available at the server, and this helps in all the applications that require the data to be available at real-time. Furthermore, if the communication link between rig site and office site is down for any reason, the data then can be requested later once the communications link is up again.

Figure A.4 demonstrates an example on WITSML Request/Subscription that should be sent by WITSML client and WITSML Response/Publication that should be sent from WITSML server back to the client with sensors data. The clients or the third-party clients have the responsibility to read the sensors data from WITSML response. Extra meta information required by the clients must be sent with the query to the server, for example: if the clients needs to get the unit of measurement for each sensor data, it should send with the query the XML tag of unit of measurements $< uom/ >$, the same can be applied on whatever required information, for more information on WITSML Standards please review [41].

A.5 WITSML Bridge Internal Design

Figure A.5 represents the internal design of WITSML bridge. The sequence of actions are executed in order to register a subscription at WITSML server for sensors data and receive the responses as publications from WITSML server.

The WITSML Bridge starts with establishing a connection to WITSML server, if the connection is successfully established then a subscription with list of sensors will be sent

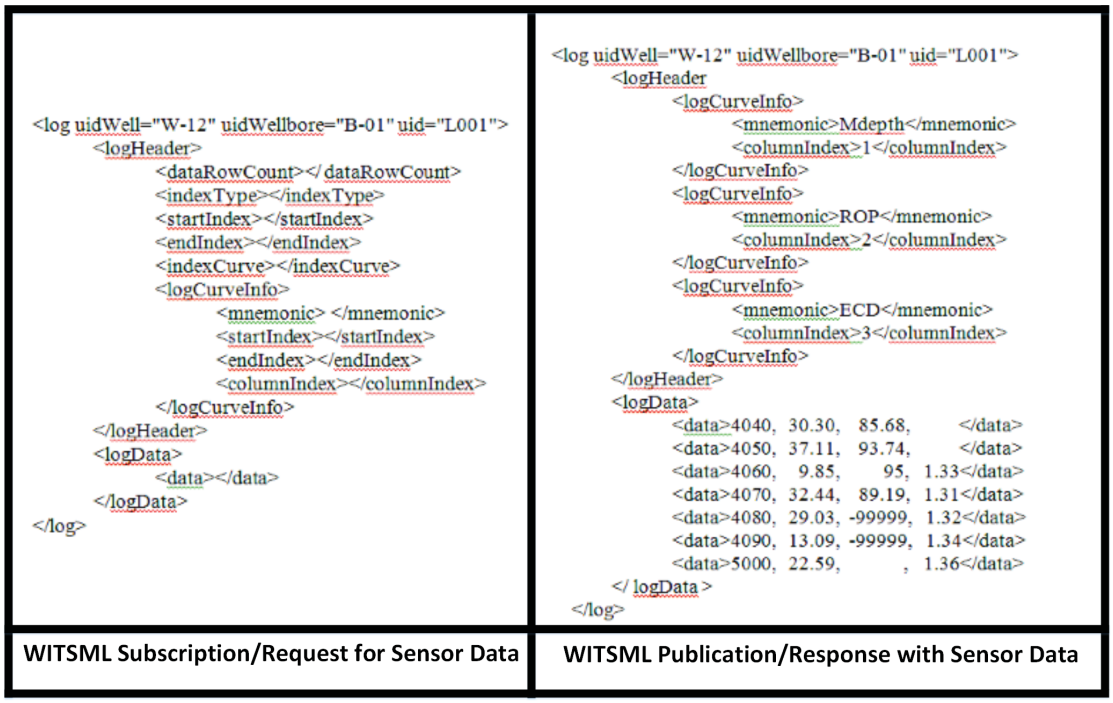


FIGURE A.4: WITSML Subscription/Request and Publication/Response

to the WITSML server, the server then will receive the request and and process it in order to prepare the results of request and send them back to WITSML bridge once the sensors data is available at the server from measurements system (see figure A.3 for more information). Once the WITSML bridge receives the response with sensors data from WITSML server, it parses the response and validates it, then the sensors data will be stored locally or forwarded to other interested parties or components (see chapter 10 for more information on how the sensors data will be processed by other components).

A.6 Experimental Results - Live Rig Data Sets

WITSML Bridge is built in order to acquire sensors data from rig sites using WITSML protocol. Five test rigs with different WITSML servers were used to test the suggested WITSML Bridge. The rigs were distributed over different parts of the world (two rigs in Norway, two rigs in North Sea, and one rig in India). The rigs have different types: three drilling platforms, one drilling Jackup, and one drilling ship. For each rig, sensors data during drilling one well is fetched, the total size of sensor data is around 200 days of drilling, the sensor data of hookload, torque, rpm, flowin, mdbit, mdhole, block position and pumps pressure were imported.

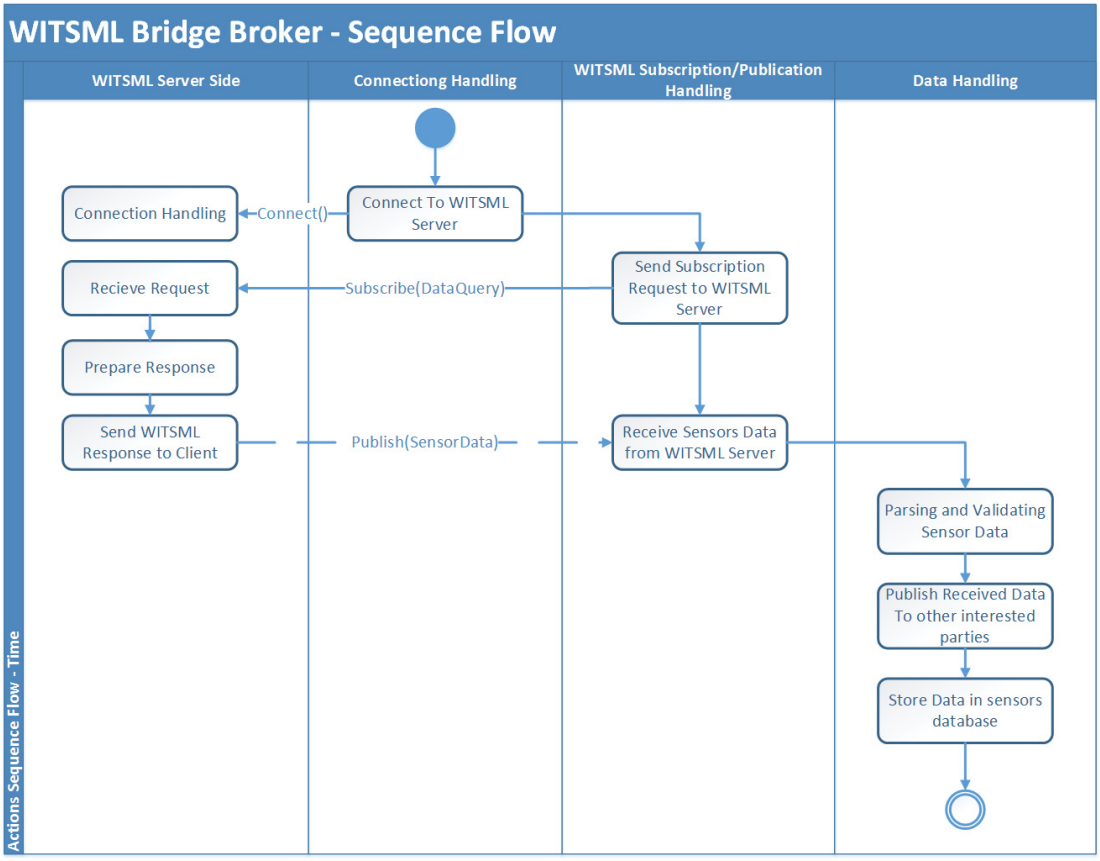


FIGURE A.5: WITSML Bridge Broker - Sequence Flow

Data set of Rig3-Well3 took around 3 months to be imported. Figure A.6 shows a summary on the five experiments, the type of each rig, frequency of data and length of sensor data.

Rig	Well	Rig Type	Location	Sensor Data Frequency	Data Length
Rig 1	Well 1	Drilling Platform -	Norway	1 second 1 Hz	12.14 days of drilling
Rig 2	Well 2	Drilling Platform	Norway	5 second 0.2 Hz	26 days of drilling
Rig 3	Well 3	Drilling Jackup	North Sea	10 second 0.1 Hz	96 days of drilling
Rig 4	Well 4	Drilling Ship	India	10 second 0.1 Hz	40 days of drilling
Rig 5	Well 5	Drilling Platform	North Sea	10 second 0.1 Hz	25 days of drilling

FIGURE A.6: Test Data Sets.

Figure A.7 displays one month sketch of sensors data from Jackup rig at North Sea $Rig3 - Well3$.

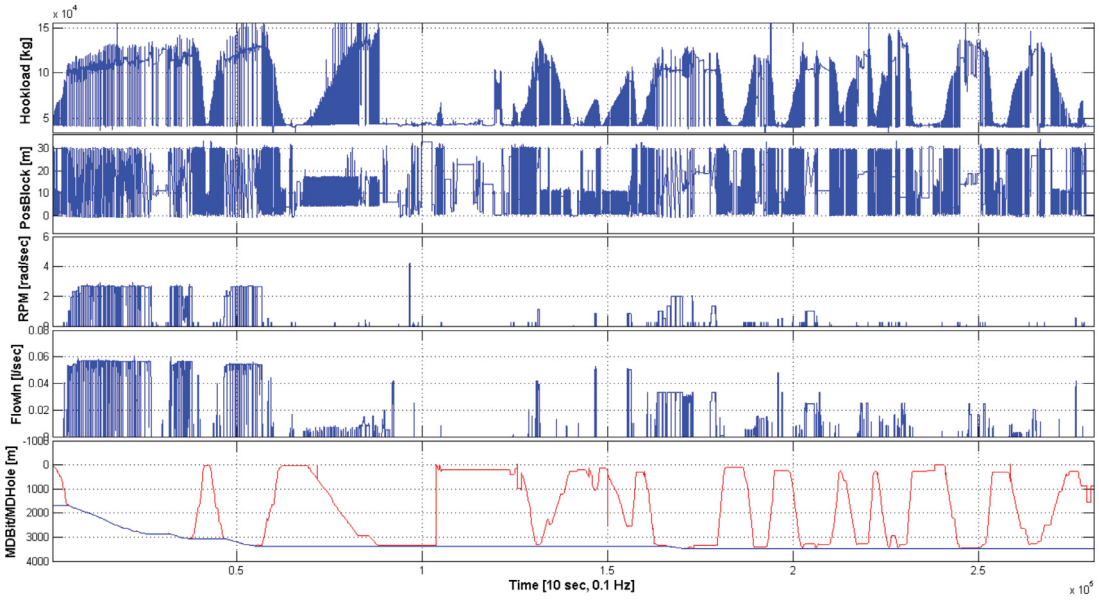


FIGURE A.7: Sensors data imported from rig for complete drilled well using WITSML Bridge.

In order to test the performance of WITSML Bridge, the calculation of completeness for each data set is done. The calculations based on ratio of the number of expected data points to the number of received data points. The results summarized in figure A.8. WITSML Bridge shows high degree of trust to fetch the data from rig site with minimum completeness around 92.5% and maximum ration of 99.5%, the total completeness ratio is 97.78%. The main reason of high completeness ratio of WITSML Bridge is that the communication link between rig site and office is a reliable link and the protocol of TCP/IP which is used as under laying protocol for communication is a reliable protocol as well. The missing data is due to problems in measurements systems of sensor malfunction at rig site.

Rig	Well	Frequency (Sec)	Data Length (days)	Expected Data Points (#)	Fetched Data Points (#)	Completeness Ratio
Rig 1	Well 1	1	12.14	1,048,896	1,013,564	96.63%
Rig 2	Well 2	5	26	449,280	445,687	99.20%
Rig 3	Well 3	10	96	829,440	824,638	99.42%
Rig 4	Well 4	10	40	345,600	341,267	98.75%
Rig 5	Well 5	10	25	216,000	199,823	92.51%
Total			**199.14**	**2,889,216**	**2,824,979**	**97.78%**

FIGURE A.8: Completeness ratios of fetched sensors data set from test rigs.

A.7 Summary

In this appendix, distributed data acquisition service called WITSML Bridge is developed in order to fetch the data from rig site. WITSML Bridge is designed to run with WITSML protocol in data formatting and web-enabled service in data communication. WITSML Server behaviour is studied and a special WTIML query language was used to query WITSML server for sensor data.

Five experiments were performed to fetch sensors data from five different rigs distributed over remote locations in the world. The WITSML Bridge shows high degree of completeness in fetching sensor data from rig sites, the performance of the bridge can be improved by tracking the gaps in fetched data set and requests back later from the server, this is in case that the reason of the gap is a failure in communication link.

Appendix B

Published Work

Journal Publications

- **Arghad Arnaout**, Bilal Esmael, Paul O'Leary, and Gerhard Thonhauser. Distributed Recognition System for Drilling Events Detection and Classification. *International Journal of Hybrid Intelligent Systems (IJHIS), ISSN 1448-5869*. 2013.
- Esmael, Bilal and **Arnaout, Arghad** and Fruhwirth, Rudolf K and Thonhauser, Gerhard. A Statistical Feature-Based Approach for Operations Recognition in Drilling Time Series. *International Journal of Computer Information Systems and Industrial Management Applications*. 2012.

Conference Publications

- **Arnaout, Arghad** and Thonhauser, Gerhard and Esmael, Bilal and Fruhwirth, Rudolf. Intelligent Real-time Drilling Operations Classification Using Trend Analysis of Drilling Rig Sensors Data. *2012 SPE Kuwait International Petroleum Conference and Exhibition*. 2012.
- **Arnaout, Arghad** and Alsallakh, Bilal and Fruhwirth, Rudolf and Thonhauser, G and Esmael, B and Prohaska, M. Diagnosing drilling problems using visual analytics of sensors measurements. *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*. 2012.
- **Arnaout, Arghad** and Esmael, Bilal and Fruhwirth, Rudolf K and Thonhauser, Gerhard. Drilling events detection using hybrid intelligent segmentation algorithm. *12th International Conference on Hybrid Intelligent Systems (HIS)*. 2012.

- **Arnaout, Arghad** and Esmael, Bilal and Fruhwirth, Rudolf K and Thonhauser, Gerhard. Model-Based Hookload Monitoring and Prediction at Drilling Rigs using Neural Networks and Forward-Selection Algorithm. *EGU General Assembly Conference Abstracts*. 2012.

- **Arnaout, Arghad** and Esmael, Bilal and Fruhwirth, Rudolf K and Thonhauser, Gerhard. Automatic threshold tracking of sensor data using Expectation Maximization algorithm. *11th International Conference on Hybrid Intelligent Systems (HIS)*. 2011.

- **Arnaout, Arghad** and Esmael, Bilal and Fruhwirth, Rudolf and Thonhauser, Gerhard. Abnormal Oil Well Drilling Operations Detection Using Smallest Principal Components. *Proceedings of 2010 The 3rd International Conference on Computational Intelligence and Industrial Application (Volume 5)*. 2010.

- Esmael, Bilal and **Arnaout, Arghad** and Fruhwirth, Rudolf K and Thonhauser, Gerhard. Automated system for drilling operations classification using statistical features. *11th International Conference on Hybrid Intelligent Systems (HIS)*. 2011.

- Esmael, Bilal and Fruhwirth, R and **Arnaout, A** and Thonhauser, Gerhard. Operations Recognition at Drill-Rigs. *EGU General Assembly Conference Abstracts*. 2012.

- Esmael, Bilal and **Arnaout, Arghad** and Fruhwirth, Rudolf K and Thonhauser, Gerhard. A hybrid multiple classifier system for recognizing usual and unusual drilling events. *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*. 2012.

- Esmael, Bilal and **Arnaout, Arghad** and Fruhwirth, Rudolf K and Thonhauser, Gerhard. Multivariate time series classification by combining trend-based and value-based approximations. *Computational Science and Its Applications–ICCSA 2012*. 2012.

- Esmael, Bilal and **Arnaout, Arghad** and Fruhwirth, Rudolf K and Thonhauser, Gerhard. Improving time series classification using Hidden Markov Models. *12th International Conference on Hybrid Intelligent Systems (HIS)*. 2012.

- Esmael, Bilal and **Arnaout, Arghad** and Fruhwirth, Rudolf K and Thonhauser, Gerhard. Automated Operations Classification using Text Mining

Techniques. *Proceedings of 2010 The 3rd International Conference on Computational Intelligence and Industrial Application (Volume 5)*. 2010.

Bibliography

[1] Fred Florence and Fionn Iversen. Real-time models for drilling process automation: Equations and applications. In *IADC/SPE Drillng Conference and Exhibition*, 2010.

[2] Tiago C Fonseca, José Ricardo Pelaquim Mendes, Adriane BS Serapiao, and Ivan Rizzo Guilherme. A genetic neuro-model reference adaptive controller for petroleum wells drilling operations. In *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, pages 3–3. IEEE, 2006.

[3] Adriane BS Serapiao, Rogerio M Tavares, Jose Ricardo P Mendes, and Ivan R Guilherme. Classification of petroleum well drilling operations using support vector machine (svm). In *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, pages 145–145. IEEE, 2006.

[4] Adriane BS Serapião, José RP Mendes, and Kazuo Miura. Artificial immune systems for classification of petroleum well drilling operations. In *Artificial Immune Systems*, pages 47–58. Springer, 2007.

[5] Adriane BS Serapião and José Ricardo P Mendes. Classification of petroleum well drilling operations with a hybrid particle swarm/ant colony algorithm. In *Next-Generation Applied Intelligence*, pages 301–310. Springer, 2009.

[6] Wolfgang Mathis and Gerhard Thonhauser. Automated reporting using rig sensor data enables superior drilling project management. In *SPE Annual Technical Conference and Exhibition*, 2006.

[7] Wolfgang Mathis, Gerhard Thonhauser, Gerhard Wallnoefer, and Johannes Ettl. Use of real-time rig sensor data to improve daily drilling reporting,

benchmarking and planning-a case study. In *Intelligent Energy Conference and Exhibition*, 2006.

[8] Wolfgang Mathis and Gerhard Thonhauser. Mastering real-time data quality control-how to measure and manage the quality of (rig) sensor data. In *SPE/IADC Middle East Drilling and Technology Conference*, 2007.

[9] Samad Valipour Shokouhi, Pål Skalle, and Agnar Aamodt. An overview of case-based reasoning applications in drilling engineering. *Artificial Intelligence Review*, pages 1–13, 2011.

[10] Samad Valipour Shokouhi, Agnar Aamodt, and Pål Skalle. A semi-automatic method for case acquisition in cbr, a study in oil well drilling. In *Proceedings of the tenth IASTED international conference on artificial intelligence and applications, AIA-10, ACTA Press, Innsbruck, Austria*, pages 263–270, 2010.

[11] Odd Erik Gundersen, Frode Sørmo, Agnar Aamodt, and Pål Skalle. A real-time decision support system for high cost oil-well drilling operations. In *IAAI*, 2012.

[12] Paal Skalle, Jostein Sveen, and Agnar Aamodt. Improved efficiency of oil well drilling through case based reasoning. In *PRICAI 2000 Topics in Artificial Intelligence*, pages 712–722. Springer, 2000.

[13] Agnar AAMODT, Pal SKALLE, Odd Erik GUNDERSEN, Frode SORMO, Jorgen SOLSTAD, et al. A method and system for monitoring a drilling operation, September 24 2010. WO Patent 2,010,106,014.

[14] George F Coulouris. *Distributed Systems: Concepts and Design, 4/e*. Pearson Education India, 2009.

[15] Greg R Andrews. *Foundations of parallel and distributed programming*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[16] Alan N Steinberg, Christopher L Bowman, and Franklin E White. Revisions to the jdl data fusion model. In *AeroSense'99*, pages 430–441. International Society for Optics and Photonics, 1999.

[17] David L Hall and James Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997.

[18] Mark Bedworth and Jane O'Brien. The omnibus model: a new model of data fusion? *IEEE Aerospace and Electronic Systems Magazine*, 15(4):30–36, 2000.

[19] Stelios CA Thomopoulos. Sensor integration and data fusion. In *1989 Advances in Intelligent Robotics Systems Conference*, pages 178–191. International Society for Optics and Photonics, 1990.

[20] Ren C Luo and Michael G Kay. Multisensor integration and fusion: issues and approaches. In *1988 Orlando Technical Symposium*, pages 42–49. International Society for Optics and Photonics, 1988.

[21] LF Pau. Sensor data fusion. *Journal of Intelligent and Robotic Systems*, 1 (2):103–116, 1988.

[22] CJ Harris, A Bailey, and TJ Dodd. Multi-sensor data fusion in defence and aerospace. *The Aeronautical Journal*, 102(1015):229–244, 1998.

[23] J Boyd. A discourse on winning and losing tech report mu 43947, 1987.

[24] Jeff Schoess and Glen Castore. A distributed sensor architecture for advanced aerospace systems. In *1988 Orlando Technical Symposium*, pages 74–87. International Society for Optics and Photonics, 1988.

[25] Belur V Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, 1997.

[26] Jaime Esteban, Andrew Starr, Robert Willetts, Paul Hannah, and Peter Bryanston-Cross. A review of data fusion models and architectures: towards engineering guidelines. *Neural Computing & Applications*, 14(4):273–281, 2005.

[27] Steve Vinoski. An overview of middleware. In *Reliable Software Technologies-Ada-Europe 2004*, pages 35–51. Springer, 2004.

[28] Xian-He Sun and Alan R Blatecky. Middleware: the key to next generation computing. *Journal of parallel and distributed computing*, 64(6):689–691, 2004.

[29] Alex Berson. *Client/server architecture*. McGraw-Hill, Inc., 1996.

[30] Robert Orfali and Dan Harkey. *CLIENT/SERVER PROGRAMMING WITH JAVA AND CORBA, (With CD)*. John Wiley & Sons, 2007.

[31] Christian Stumm. Client-server system for delivery of online information, June 16 1998. US Patent 5,768,528.

[32] Matteo Bertocco, Franco Ferraris, Carlo Offelli, and Marco Parvis. A client-server architecture for distributed measurement systems. *Instrumentation and Measurement, IEEE Transactions on*, 47(5):1143–1148, 1998.

[33] Valeria Cardellini, Michele Colajanni, and Philip S Yu. Dynamic load balancing on web-server systems. *Internet Computing, IEEE*, 3(3):28–39, 1999.

[34] Haakon Bryhni, Espen Klovning, and Oivind Kure. A comparison of load balancing techniques for scalable web servers. *Network, IEEE*, 14(4):58–64, 2000.

[35] Daniel M Dias, William Kish, Rajat Mukherjee, and Renu Tewari. A scalable and highly available web server. In *Compcon'96.'Technologies for the Information Superhighway'Digest of Papers*, pages 85–92. IEEE, 1996.

[36] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. *A scalable content-addressable network*, volume 31. ACM, 2001.

[37] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web services*. Springer, 2004.

[38] Hugo Haas and Allen Brown. Web services glossary. *W3C Working Group Note (11 February 2004)*, 2004.

[39] Mike Botts, George Percivall, Carl Reed, and John Davidson. Ogc® sensor web enablement: Overview and high level architecture. In *GeoSensor networks*, pages 175–190. Springer, 2008.

[40] MA Kirkman, ME Symmonds, SW Harbinson, JA Shields, M Will, and A Doniger. Wellsite information transfer standard markup language, witsml, an update, 2003.

[41] *WITSML Standards*.

[42] Ken Birman and Thomas Joseph. *Exploiting virtual synchrony in distributed systems*, volume 21. ACM, 1987.

[43] Peter R Pietzuch and Jean M Bacon. Hermes: A distributed event-based middleware architecture. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pages 611–618. IEEE, 2002.

[44] Douglas C Schmidt. Middleware for real-time and embedded systems. *Communications of the ACM*, 45(6):43–48, 2002.

[45] Wendi B Heinzelman, Amy L Murphy, Hervaldo S Carvalho, and Mark A Perillo. Middleware to support sensor network applications. *Network, IEEE*, 18(1):6–14, 2004.

[46] Sacha Krakowiak. Middleware architecture with patterns and frameworks. 2007.

[47] Chris Chatfield. *The analysis of time series: an introduction*. CRC press, 2003.

[48] T Warren Liao. Clustering of time series dataa survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[49] Rui Xu, Donald Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[50] Thomas H Wonnacott and Ronald J Wonnacott. *Introductory statistics*, volume 19690. Wiley New York, 1972.

[51] Xinhua Zhuang, Yan Huang, Kannappan Palaniappan, and Yunxin Zhao. Gaussian mixture density modeling, decomposition, and applications. *Image Processing, IEEE Transactions on*, 5(9):1293–1302, 1996.

[52] BS Everitt, S Landau, and M Leese. Cluster analysis arnold. *A member of the Hodder Headline Group, London*, 2001.

[53] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[54] Mario A. T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(3):381–396, 2002.

[55] Geoffrey McLachlan and David Peel. *Finite mixture models*. Wiley. com, 2004.

[56] THRIYAMBAKAM Krishnan and GJ McLachlan. The em algorithm and extensions, 1997.

[57] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002.

[58] Gilles Celeux and Gérard Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3): 315–332, 1992.

[59] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and Information Systems*, 8(2):154–177, August 2004. ISSN 0219-1377. doi: 10.1007/s10115-004-0172-7. URL http://link.springer.com/10.1007/s10115-004-0172-7.

[60] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[61] Hagit Shatkay and Stanley B Zdonik. Approximate queries and representations for large data sequences. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, pages 536–545. IEEE, 1996.

[62] Huanmei Wu, Betty Salzberg, and Donghui Zhang. Online event-driven subsequence matching over financial data streams. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 23–34. ACM, 2004.

[63] Oliver Amft, Holger Junker, and Gerhard Troster. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 160–163. IEEE, 2005.

[64] Pierre Geurts. Pattern extraction for time series classification. In *Principles of Data Mining and Knowledge Discovery*, pages 115–127. Springer, 2001.

[65] Eamonn J Keogh and Michael J Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD*, volume 98, pages 239–243, 1998.

[66] Chotirat Ann Ratanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michail Vlachos, and Gautam Das. Mining time series data. In *Data Mining and Knowledge Discovery Handbook*, pages 1049–1077. Springer, 2010.

[67] David Sankoff and Joseph B Kruskal. Time warps, string edits, and macromolecules: the theory and practice of sequence comparison. *Reading: Addison-Wesley Publication, 1983, edited by Sankoff, David; Kruskal, Joseph B.*, 1, 1983.

[68] Arghad Arnaout, Bilal Esmael, Rudolf K Fruhwirth, and Gerhard Thonhauser. Drilling events detection using hybrid intelligent segmentation algorithm. In *Hybrid Intelligent Systems (HIS), 2012 12th International Conference on*, pages 508–511. IEEE, 2012.

[69] Arghad Arnaout, Bilal Esmael, Rudolf K Fruhwirth, and Gerhard Thonhauser. Automatic threshold tracking of sensor data using expectation maximization algorithm. In *Hybrid Intelligent Systems (HIS), 2011 11th International Conference on*, pages 551–554. IEEE, 2011.

[70] Hussain Rabia. *Oilwell drilling engineering: principles and practice*. Graham & Trotman, 1985.

[71] G Thonhauser. Using real-time data for automated drilling performance analysis. *OIL GAS European Magazine, Edition*, 4:170–173, 2004.

[72] Douglas Reynolds. Gaussian mixture models. *Encyclopedia of Biometric Recognition*, 2(17.36):14–68, 2008.

[73] Hubert W Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62 (318):399–402, 1967.

[74] John A Rice. *Mathematical statistics and data analysis*. Cengage Learning, 2007.

[75] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, September 1999. ISSN 0360-0300. doi: 10.1145/331499.331504. URL http://doi.acm.org/10.1145/331499.331504.

[76] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[77] Frank Dellaert. The expectation maximization algorithm. 2002.

[78] E Weinstein. Expectation maximization algorithm and applications. 2006.

[79] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[80] Takio Kurita, Nobuyuki Otsu, and N Abdelmalek. Maximum likelihood thresholding based on population mixture models. *Pattern Recognition*, 25 (10):1231–1240, 1992.

[81] Alan H Fielding and John F Bell. A review of methods for the assessment of prediction errors in conservation presence/absence models. *Environmental conservation*, 24(1):38–49, 1997.

[82] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[83] Paul OLeary Arghad Arnaout, Bilal Esmael and Gerhard Thonhauser. Distributed recognition system for drilling events detection and classification. *International Journal of Hybrid Intelligent Systems (IJHIS), ISSN 1448-5869*, 2013.

[84] Sahibsingh A Dudani, Kenneth J Breeding, and Robert B McGhee. Aircraft identification by moment invariants. *Computers, IEEE Transactions on*, 100 (1):39–46, 1977.

[85] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[86] Thomas H. Reiss. The revised fundamental theorem of moment invariants. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(8): 830–834, 1991.

[87] Paul OLeary and Matthew Harker. Surface modelling using discrete basis functions for real-time automatic inspection. *Journal of Electronic Imaging*, 18, 2010.

[88] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and image processing*, 14(3):227–248, 1980.

[89] Jaime G Carbonell, Ryszard S Michalski, and Tom M Mitchell. An overview of machine learning. In *Machine learning*, pages 3–23. Springer, 1983.

[90] John Wiley. Ieee programs for digital signal processing. (8), 1979.

[91] RE Jantzen, RD Foreman, L Keltner, and RS McCoy. Wellsite information transfer specification (wits) for digital rig-site data. *SPE drilling engineering*, 4(4):291–299, 1989.

[92] Nigel Deeks and Thomas Halland. Witsml changing the face of real-time. In *Intelligent Energy Conference and Exhibition*, 2008.

[93] Musab Al-Khudiri, Majid Al Shehry, and JD Curtis. Data architecture of real-time drilling and completions information at one company. In *SPE Russian Oil and Gas Technical Conference and Exhibition*, 2008.