Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Identifying Operator Usage of Wheel Loaders Utilizing Pattern Recognition Techniques

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan vid Linköpings universitet
av

Karin Ohlsson-Öhman

LiTH-ISY-EX--12/4591--SE

Linköping 2012

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Identifying Operator Usage of Wheel Loaders Utilizing Pattern Recognition Techniques

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan i Linköping
av

Karin Ohlsson-Öhman

LiTH-ISY-EX--12/4591--SE

Handledare:	**Tomas Nilsson**
	ISY Linköpings universitet
	Bobbie Frank
	Volvo CE
Examinator:	**Erik Frisk**
	ISY, Linköpings universitet

Linköping, 14 June, 2012

Avdelning, Institution
Division, Department

Division of Automatic Control
Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Datum
Date

2012-06-14

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ ______________

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ ______________

ISBN
—

ISRN
LiTH-ISY-EX--12/4591--SE

Serietitel och serienummer
Title of series, numbering

ISSN
—

URL för elektronisk version

`http://www.control.isy.liu.se`
`http://www.ep.liu.se`

Titel
Title Identifying Operator Usage of Wheel Loaders Utilizing Pattern Recognition Techniques

Författare
Author Karin Ohlsson-Öhman

Sammanfattning
Abstract

The information about how wheel loaders are used by costumers is not well-documented, but large quantities of sensor signal data are recorded. This data makes the starting point to develop a robust automatic pattern recognition algorithm to identify repetitive driving operations.

The algorithm is composed by models of events that appear in cycles. The cycles are pre-defined as short loading cycle and load and carry. They consist of the phases; load, drive loaded, unload and drive unloaded. In the load and carry cycle, the driving forward loaded phase can last up to 400 meters. Beyond these two is something called sub cycle defined. It contains cleaning cycle and reloading. A cycle is identified by transition automata of events. To make the identification more robust against disturbances, a probability function is connected to the cycle identification.

The developed algorithm uses signals from sensors available in series production wheel loaders. These signals are used to identify cyclic behaviour in the shape of short loading cycle, load and carry and sub cycle. After removal of completely standing still events, the algorithm calculates the characteristic parameters for the identified cycles and for their phases and present them in an excel file. Validation showed that the algorithm can find 75 % or more of cyclic behaviour for bucket handling.

Nyckelord
Keywords Wheel loaders, pattern recognition

Abstract

The information about how wheel loaders are used by costumers is not well-documented, but large quantities of sensor signal data are recorded. This data makes the starting point to develop a robust automatic pattern recognition algorithm to identify repetitive driving operations.

The algorithm is composed by models of events that appear in cycles. The cycles are pre-defined as short loading cycle and load and carry. They consist of the phases; load, drive loaded, unload and drive unloaded. In the load and carry cycle, the driving forward loaded phase can last up to 400 meters. Beyond these two is something called sub cycle defined. It contains cleaning cycle and reloading. A cycle is identified by transition automata of events. To make the identification more robust against disturbances, a probability function is connected to the cycle identification.

The developed algorithm uses signals from sensors available in series production wheel loaders. These signals are used to identify cyclic behaviour in the shape of short loading cycle, load and carry and sub cycle. After removal of completely standing still events, the algorithm calculates the characteristic parameters for the identified cycles and for their phases and present them in an excel file. Validation showed that the algorithm can find 75 % or more of cyclic behaviour for bucket handling.

Acknowledgments

I would like to thank my supervisor, Bobbie Frank, at Volvo for the great opportunity to do my thesis work there, and for the help and ideas. Thanks to Magnus Larson and Lennart Skoogh at Volvo for measurement data and answer to my software questions.

I would also like to thank my examiner Erik Frisk and my supervisor Tomas Nilsson at Fordonssystem, Linköping University.

A special thanks to my family for their great support, Andreas Midestad for your support and help. Thanks to Sandra Alfredsson that cheered me on and corrected language in the report.

Karin Ohlsson-Öhman
Linköping 2012

Contents

Chapter 1

Introduction

This report presents the work made in the master thesis, Identifying Operator Usage of Wheel Loaders Utilizing Pattern Recognition Technique. The object was to identify repetitive usage of wheel loaders utilizing pattern recognition on signals from sensors available in series production wheel loaders. This chapter gives an introduction in shape of a background, a quick introduction to wheel loader usage, the objectives of the thesis work and a short review of previous and related research.

1.1 Background

Volvo Construction Equipment has an interest to get better knowledge about how their costumers use wheel loaders, to increase the possibilities for doing advanced adaptive online machine control strategies.Wheel loaders are used in many different applications ranging from snow ploughing to bucket handling. Experience shows that right choice of configuration on the machine have a huge effect on the efficiency. Results in the paper [1] indicates that approximately 10 % of the fuel cost could be reduced with the right control strategies. Unfortunately there exist no basic data put together with information about handling cycles and material. Instead, rough estimations are made, which make it hard to make customer specific applications for the machines. But there exist large quantities of signal data from sensors available in series production wheel loaders from for example the control system. The sensors can for example measure pressure, velocity and temperature. The master thesis has a starting position in this collected data together with the previous research described next.

1.1.1 Previous research

In a research project between Volvo CE and the department of Electrical Engineering, Linköping University an algorithm that identifies repetitive pattern has been developed. The research is presented in the unpublished article ”Driving patter detection and identification under usage disturbances” [2]. This research is the starting point for the thesis and in this section the article is summarized and the results are briefly discussed.

An automatic algorithm has been developed in for identifying operating cycles only using sensors signal available in series production wheel loaders. Figure 1.1 shows a configuration of the vehicle with sensor placement.

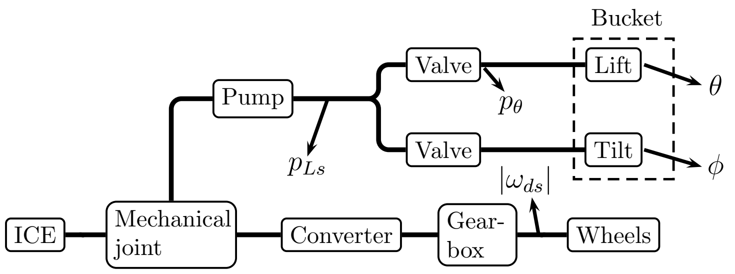


Figure 1.1: Schematic view over the configuration of the vehicle. Pressure in the lift cylinder, p_θ, Ls pressure, p_{Ls}, lift angle, θ, tilt angle, ϕ and angular speed of the drive shaft is denoted ω_{ds}. Note only p_θ is not from a production sensor.

The algorithm is designed to detect and identify predefined cycle patterns in unstructured data. The cycles can be divided in phases and events. The events are defined from characteristic operations, the events are:

- loading, l
- start driving forward, f
- start driving in reverse, r
- unload, u

Each one of the events is specified by conditions and is identified when the conditions are fulfilled. The driving events, forward and reverse, identifies when a transition in direction of the gear lever is made. The criteria for a driving forward event is that the gear lever should be changed from reverse to forward. Reverse driving events should fulfill the reverse criteria. Note that the event only registrates once per transition.

The model for bucket loading contains conditions on lift- and tilt angle and also driving direction. For a bucket loading event to be fulfilled requires that both the lift angle θ and the tilt angle ϕ significantly increases over a specified time interval and in the same time the vehicle is moving forward. Equations for the conditions can be seen in equation 1.1 that is collected from the paper [2].

$$l_{k-L} \; is \; generated \; if \; \theta_k - \theta_{k-L} > \alpha \; and \; \phi_k - \phi_{k-L} > \beta \; and \; v_{k-L} > 0 \tag{1.1}$$

where α and β are detector parameters and L is the length of the time interval. When all the conditions are fulfilled the loading event is identified and given a time-stamp same as the start time in the time interval. Note that a loading event can be identified multiple times during one bucket loading.

The model for unloading is simpler than the model for loading, it only treats the tilt angle. The condition is fulfilled when the tilt angle ϕ is small enough, in equation 1.2 [2] this can be seen.

$$u_k \ is \ generated \ if \ \phi_{k-1} >= \xi \ and \ \phi_k < \xi \tag{1.2}$$

where ξ is a design parameter. In the same way as loading events unloading events are given a time-stamp. Several unloading events can be detected after each other but for this to happen requires that the operator once again is emptying the bucket.

When all the events are identified they are sorted after the time-index with the smallest index first. Sequences of events that are identified by a state automata, figure 1.2 are called cycles. Cycles are pre-defined according to the description in section 1.1.2. A transition diagram for identification of a loading cycle can be seen in figure 1.2.

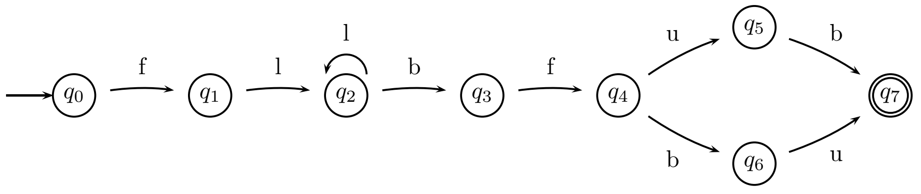


Figure 1.2: Transition diagram over the automata for cycle identification [2], where the transitions are events.

A schematic view of a cycle can be seen in figure 1.3. The cycle can be divided in time-intervals called phases. There are seven phases, they are listed below:

- forward motion
- backward motion
- direction change
- bucket filling
- bucket loading
- bucket emptying
- bucket unloading

From figure 1.3 all the phases can be described. Start with forward motion, the wheel loader drives forward from point 2 to point 3 is a forward motion phase and also when the vehicle is driving from point 2 to point 1. Backward motion is driven between point 1 and point 2 and also from point 3 to point 2. The phases bucket filling and bucket loading are made at point 1 and the phases bucket emptying and bucket unloading are made at point 3. The seven phases are divided in to two sequences, vehicle motion and loading handling. The reason why the phase portioning is made after the cycle identification is to make the algorithm more robust.

After the cyclic behaviour has been identified interesting parameters are calculated. Examples of parameters are vehicle speed and gear selection. The cycles are also categorized as long or short loading cycle. A bucket load estimator is also developed according to a model of the relations between the pressure in the lift cylinder and the mass using machine geometry.

1.1.2 Wheel loader usage

A wheel loader is a versatile machine and is built in different sizes and have different working areas and attachments. This results in that basically every work site is unique, nevertheless it is possible to see some common features. The knowledge about wheel loader operation and cycle behaviour is mostly collected from the Ph.D thesis [3] but also from discussions and some own experience.
In this thesis three sorts of cycles are treated, they are:

- Short loading cycle
- Load and carry
- Sub cycle

This section contains a description on how the different cycles are operated and how they differ from each other. For example is the first two cycles loading cycles and in many aspects very similar.

The first one is called short loading cycle, SLC, and is representative for a majority of usage. One example where this type of cycle is used is bucket loading of some kind of material, that can be sand, gavel or shot rock, on a close load receiver, for example a articulated hauler or a stone crusher. The whole cycle lasts around 25-35 seconds. An illustration of the cycle is shown in figure 1.3.

The SLC can be divided in to phases. This thesis work uses four phases and starts with a loading phase, the wheel loader is at position 1 that can be seen in figure 1.3. The starting point for the loading cycle is assumed to be the point when the wheel loaders start driving into the pile and the lift arm is raising followed by tilt movement. The loading phase ends when the shift lever is put in reverse, this is also the start point for the first part of phase two, a transport phase with load. The wheel loader drives in reverse to point 2 in the figure, point 2 is the position where the shift lever is put to forward. The transport phase ends when the wheel

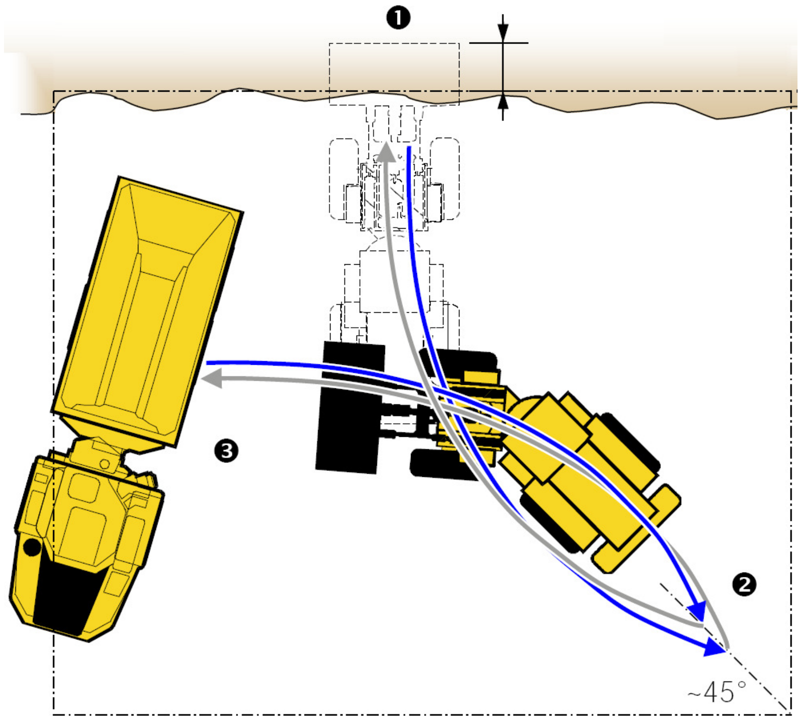


Figure 1.3: Schematic view over a short loading cycle [3].

loader is driven to point 3 and is ready to start unloading. The third phase is the unloading phase, the lift arm raises and the bucket tilts down, this phase ends when the shift lever is put in reverse. The last phase is a transport phase, but without load, it reflects the earlier transport phase, reverse driving to point 2 then drive forward until it is in position to start a new loading phase.

The other main cycle that is widely used is load and carry, LAC. It starts with a loading phase with the same execution as SLC. The rest of the phases in LAC are also the same as the phases for SLC. The main difference lies in the transport phases, where the distance can be as much as 400 meters. For this type of cycle a common receiver is a pocket that goes to a conveyor belt or a stone crusher, it is even not uncommon that the last meters of the transport has a steep inclination that the wheel loader needs to climb.

The third type of cycle that is treated is called sub cycle because it is not a preoccupation of the work that is executed. These cycles can be made a couple of times between SLC and LAC or in one of the cycles but this is only a small part of the whole usage. In this thesis cleaning of spill is when the loading sequence is finished or in the middle of the loading process categorized as a sub cycle. The cleaning cycle is made by cleaning up spill by driving forward with the bucket against the ground. When the vehicle is near the pile it raises the lift arm, drives closer and tilts the bucket so it is emptied. The cleaning cycle ends when wheel loader has driven in reverse and change the gear lever direction to forward. The other type of operation that is categorized as a sub cycle is reload. This type of behaviour starts as an ordinary loading phase as in SLC or LAC, but the loading fails so the operator is emptying the bucket and reverse driving for load once again. The behaviour is very similar to the cleaning cycle.

1.2 Purpose and goal

The purpose of this thesis is to modify and develop the existing algorithm described in section 1.1.1 to get it more robust against disturbances. The algorithm should be able to handle and separate bucket, pallet and timber claw as attachment. This can give a better understanding about how Volvo CE costumers use their machines and also enable advanced adaptive online machine control strategies. The algorithm can be a start for a future work to optimize control applications to increase the fuel efficiency and productivity on the wheel loader.

The existing algorithm can identify bucket cycles and divide the cycles into phases. It can also categorize the loading cycles in short loading cycle and load and carry. It can, with the use of sensors available on a series production wheel loader, estimate the load. The goal is that the algorithm only use production sensors. The algorithm should also be expanded so it can identify handling with the earlier named attachment and separate them if they appear in the same dataset. This work should also investigate which the interesting characteristic parameters are and the algorithm should present them in a perspicuous way.

It is desirable that the algorithm can identify at least 80 % of the cycle behaviour in a dataset. For validation of the algorithm a film from the front window is studied. Where the cycles appear and how many they are is identified manually from the film. The manual result are compared with the result from the algorithm to be able to quantify of the algorithms performance.

1.3 Methods

The idea is to continue with the same approach as the existing algorithm. That means create functions that identifies events from the way the wheel loader operates, use or modify the events that are described in section 1.1.1 and develop new. All the events are then sorted after appearance, from this cumulation of events identify patterns of repetitive behaviour by an automata. Connected to the cycle identification should a probability function be attached. The probability function should handle disturbances in the way the events occur to increase the cycle identification performance.

Another feature to increase the robustness that should be implemented is a pre algorithm that applies signal processing on the signals. The pre algorithm should also single out the signals that are needed and save them in a temporary file that should be deleted in the end of the algorithm.

The algorithm should also be developed in a user friendly way in . In the beginning of the existing algorithm it is very easy to just type run main for start the algorithm and then chose which dataset to run. This will expand with a possibility to type in information that is missing and choose if characteristic parameters should be stored in an excel file.

1.4 Related research

Research on pattern recognition is going on in many different areas. It is possible to identify patterns in speech and face recognition. Markov models are used to recognize human behaviour from sensory data when driving to predict the behaviour [4]. Hidden Markov models are used for recognition of driving events, to be capable to develop useful navigation support [5].

Other areas of research are to identify what kind of road a car is driving on by apply statistical pattern recognition framework implemented by means of feedforward neural networks, [6]. Models were also developed from vehicle control data, and the classification was made in four classes of driving situations: highway, main road, suburban traffic and city traffic. Research is made with purpose to design the control strategy to minimize fuel consumption, [7]. From driving characteristics are six driving patterns classes designed to represent different driving scenarios. A

driving pattern recognition method is used to classify the current driving pattern into a driving pattern class. For every driving pattern class dynamic programming techniques are utilized to find the optimal control actions.
For many applications, pattern recognition from multisensor time series is an important problem. Predefined templates are used for identify flight maneuver for multisensor time series are one of them, [8]. The method using templates is an alternative when not enough training and testing data for other identification scheme exist.

1.5 Outline of the master thesis

The master thesis is outlined according to:

Chapter 1: Introduction to the area of the master thesis and a description of the ambition of the thesis work and expected result. A description of common uses of wheel loaders and related research are also given.

Chapter 2: Short theory over methods that can be used for pattern recognition.

Chapter 3: The algorithm part is described and how it works.

Chapter 4: Results and conclusions from the thesis work are presented here.

Chapter 5: A discussion over difficulties in the work is held here. Some ideas and thoughts about development of the algorithm and reasonable alternative ways to solve found problems are presented.

Chapter 6: Gives suggestions of improvements and future work with the driving pattern recognition algorithm.

Appendix A: Table of calculated characteristic parameters.
Appendix B: Table of produced plots.
Appendix C: User guide for the algorithm.

Chapter 2

Theory

Pattern recognition is most commonly used for recognising patterns in images. An example is to recognize faces in images. Other areas where pattern recognition is studied are psychology, ethology, traffic flow and computer science. Methods in the area of pattern recognition that are related to this thesis are described in this chapter.

2.1 Pattern recognition methods

There are many methods for pattern recognition, two large areas, Markov Chain Monte Carlo and neural networks, and two small areas, temporal reasoning and automata, are briefly presented.

2.1.1 Markov Chain Monte Carlo techniques

A powerful framework that allows sampling from a large class of distribution is called Markov Chain Monte Carlo (MCMC), [9]. A first-order Markov chain is by definition a series of random variables $z^{(1)}, ..., z^{(m)}$ that hold conditional independent properties according to formula (2.1).

$$p(z^{(m+1)}|z^{(1)}, ..., z^{(m)}) = p(z^{(m+1)}|z^{(m)}) \tag{2.1}$$

This can be represented by a directed graph, an example of that can be seen in figure 2.1.

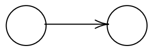


Figure 2.1: A simple example of a directed graph, the graph is directed in one way only, in the direction indicated by the arrow.

The Markov chain can be specified by giving the initial variable $p(z^{(0)})$ a probability distribution and also give the subsequent variables a conditional probability in the form of transition probabilities. If the transition probabilities are the same for all m variables, the Markov chain is called homogeneous.
A characteristic Markov chain process has the following steps [10]:

1. A finite number of possible states.
2. The process can be in one, and only one state at a time.
3. The process steps successively from one state to another over time.
4. The probability of a move depends only on the immediately preceding state.

Pattern recognition based on some form of MCMC is computational heavy [11] because of the sampling, this is a reason to not use such methods in this thesis.

2.1.2 Temporal reasoning

The phenomenon reasoning about events that depend on time is called temporal reasoning. Humans do it easily but it is harder to formalize temporal events so that a computer can make temporal inference, [10]. Aircraft traffic controller is an example of an expert system that reason about temporal events that can be useful. In medicine, expert systems that reason over time have been developed. Many of the medical systems have less difficult problem with temporal reasoning compared to an aircraft traffic controller, which must operate in real time. Systems that work in real time have difficulties to build multiple hypotheses for temporal reasoning because of the large amount of processing required and of the inference engine design. Also an expert system for real time that does a lot of temporal reasoning to explore multiple hypotheses is very difficult to build. To make a decision for any kind of used hypotheses, many different logics can be developed. Another way to approach temporal reasoning is to use probabilities. The system can be seen as moving from one state to another as evolving over time. The progression is called stochastic process. More can be read in [10] or a book about statistic theory.

Temporal reasoning is used in this thesis to find events (chapter 3.2.1) and separate different loading events.

2.1.3 Automata

An automata can be used for finding predefined patterns of events in strings [12]. An automata can identify and give index to predefined patterns. This method is used for modelling cycles, that are described in section 1.1.2, and identification of the cycle in the string of events. Automata are states, to get to the next state a condition is needed to be fulfilled. If the condition is not true then the automata needs to start from the beginning. The figure 1.2 illustrates an automata that was used in the previous research, described in section 1.1.1. Automatas are used for identification of all types of cycles, the cycle identification is described in chapter 3.2.2.

2.1.4 Neural networks

A successful model for using adaptive parameters is feed-forward neural networks. Parametric forms are used for the basic functions and the parameter values are adopted during training. A price for using this is that the likelihood function, which forms the basis for network training, is no longer a convex function of the model parameters.

The network can have more than the input layer and the output layer, the layers in-between are called hidden layers. The layers are connected, the connections are weighted and how the weight is distributed is related to the probability. Figure 2.2 shows an illustration over a neural network. The bottom layer in the figure symbolize the input layer. The input layer is connected to the hidden layers, in the figure this is symbolized with an arrow. The connection has weights that changes during training when both the input and the output data are known. In the figure there is only one hidden layer, but there can also be many hidden layers. The hidden layer is connected to the output layer (if more then one hidden layer, the last one is connected to the output layer), these connections are also weighted and illustrated with an arrow. The weights are measurements of probability for the connections between input and output. The top layer is the output layer, which returns the result from the network.

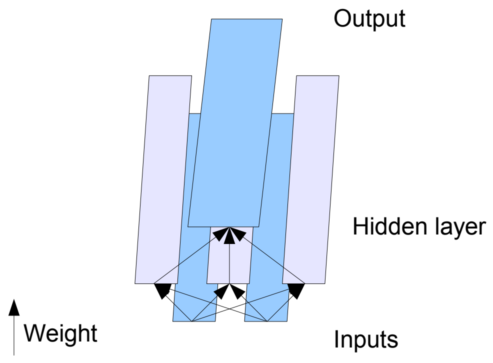


Figure 2.2: A possible look of a feed-forward neural network.

The neural network is trained by giving it an input vector and a target vector (output) with the goal to minimize the error between them. The weights, w, are given the values that minimize the error function. There are many ways to calculate the weights, [9]. When the weights are set to proper values, the network can be tested with only input data and the output should be correctly identified. The more training the network gets the better it becomes for giving the right output, note that the network can be over trained which results in incorrect weight distribution. An over trained network adjust after the data and not for the system. An example, a network should be trained to make a left or right turn, the training data mostly contains right turns. If the network is over trained it has learned to always turn right.

2.2 Other theory

Pattern recognition theories are closely related to probability theories. Basic probability theory will not be discussed here but one theorem is described.

2.2.1 Bayes' theorem

The probability of an event A, given that an event B has occurred is called a conditional probability, $p(A|B)$. The inverse probability, to find the probability of an earlier event given that a later one has occurred can be solved by Bayes' theorem, (2.2).

$$p(B|A) = \frac{p(A \cap B)}{p(A)} \tag{2.2}$$

It is possible to use the Multiplication Law, $p(A \cap B) = p(A|B)p(B)$ in equation (2.2) and then get an alternative equation, (2.3).

$$p(B|A) = \frac{p\left(A|B\right)p\left(B\right)}{p(A)} \tag{2.3}$$

The theory about Bayes' theorem [10] is used for the decision in the probability function described in chapter 3.3.

Chapter 3

Algorithm

The algorithm can be divided into three different parts, in figure 3.1 a schematic picture over the composition of the algorithm is shown. The first part is a pre algorithm where signal processing is applied to the signals used by the main algorithm, the next part in the algorithm. The main algorithm detects and identifies cyclic behaviour in the sensor signals. The algorithm ends with the post algorithm that calculates characteristic parameters in the cycles. In this chapter the sections of the algorithm are described.

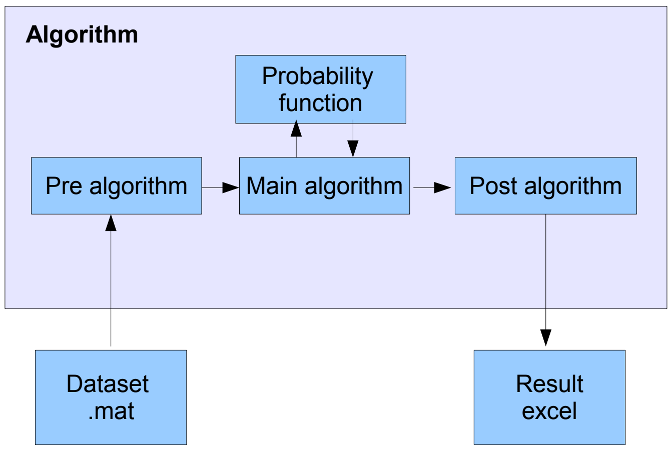


Figure 3.1: A schematic view over the algorithm structure. The way the signal data is passing through the algorithm is illustrated with arrows.

3.1 Pre algorithm

The purpose of the pre algorithm is to prepare the signals for the identification and to reduce the amount of data that is sent through the algorithm. Selected signals are picked out and saved in a temporary file that is sent further in the algorithm. The pre algorithm is one part of making the algorithm more robust by applying signal processing on the used signals. Table 3.1 summarizes the signals that are used together with a short description.

Table 3.1: Table with the signal name that the algorithm is using.

Signal name	**Description**
AcceleratorP	Accelerator pedal applied. [%]
ActualGear	Actual gear
ActualGearDi	Actual gear direction, driving forward or reverse
ActualEngine	Actual engine torque
BrakePedalAp	Brake pedal applied
EngineSpeed	Engine speed
FirstAngle	Lift angle
FirstLevPos	The lever for the lift function
freq	The highest sample frequency
LsPressure	The load sensing pump pressure
OutputTorque	Output torque
SecAngle	Signal for the tilt function
SecLevPos	Position of the lever related to the tilt function
ThirdLevPos	The third function lever
timecan	Time vector for the highest sampled signal
TurbineSpee	Signal for the velocity of the turbine
VehicleSpe	Signal for the vehicles velocity

3.1.1 Naming

The name of the signals that the algorithm is using can be seen in the left column in table 3.1. The naming of the signals varied in the used datasets which caused problems for the algorithm. To remove this problem a naming identifier has been implemented. The naming identifier first searches for a signal with the name that the rest of the algorithm uses, if it does not find the signal it searches for alternative names on the signal that have appeared in the used data. If none of these names exist, the algorithm asks the user to type in the name of the signal, if the case is that even this signal does not exist the algorithm ends.

3.1.2 Signal Processing

After the naming algorithm has found the signals, signal processing is applied. Peaks are removed and all the signals are upsampled to the highest sample frequency to make it easier to compare signals and to make the algorithm independent of the sample frequency. Two techniques are used for upsampling; linear interpolation and moving average spline interpolation.

Peak removal

The untreated signal for the wheel loader speed sometimes contains peaks from zero up to the maximum speed for one or two samples. This peculiar behavior can appear when the equipment is attached, or when the machine is turned on or off. This unexpected behavior has made the algorithm to incorrectly identify or not identify cycles. To avoid this a function in the pre algorithm has been implemented that reduces all peaks and give them the same samplings value as the previous sample. The equation for the peak removal

$$if\ v_{i-1} == 0\ and\ v_i > 45\ [km/h]\ do\ v_i = v_{i-1} \tag{3.1}$$

where v_i is the vehicle velocity at the time i. If the vehicle speed signal sample frequency is not the same as the highest samplings frequency the signal is upsampled by linear interpolation.

Moving average spline

The derivative of the lift and tilt signal are studied to decide if the angle is increasing or decreasing. The original signals are updated with a lower frequency then the actual sampling frequency, which make it piecewise constant. This makes it necessary to further process the signal to be able to reliably compute the derivative. Moving average is applied on the signals for smoothing, after that the derivative can be computed. Most of the datasets used for development, both the lift and tilt signal had the lower sample frequency and where upsampled by spline interpolation. In figure 3.2 the lift (dotted) and tilt (continuous) can be seen. In the upper plot the signals are untreated and in the lower plot are the same signals after the smoothing function has been applied and it is easy to see improvement in the signals.

A small time shift of the signals does not effect the algorithm because both the signals are equally shifted. The possible time displacement is small so the position where loading and unloading event is identified does not differ from the actual position. More about event identification can be read in section 3.2.1.

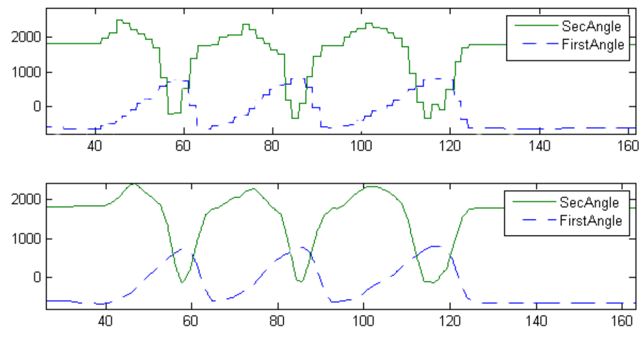


Figure 3.2: The upper plot shows the lift angle (first angle) and tilt angle (second angle) untreated and in the lower plot shows the same signals processed by moving average.

Vehicle speed

The true vehicle velocity is calculated from the drive shaft angular speed $|\omega_{ds}|$ and the information about the gear selection. The gear is transformed to be negative when the vehicle direction is backwards and positive when the direction is forward. This is multiplicated with the vehicle speed and the result is a true vehicle speed that is negative when the wheel loader is travelling backwards and positive for forward driving.

3.2 Main algorithm

This part of the algorithm identifies cyclic behaviour, the cycles are predefined and described in section 1.1.2. The identification is made in three parts;

- Events
- Cycles
- Phases

Events are distinct wheel loader operations, without any expansion in time, such as loading and start driving forward. From the events the predefined cycles are identified based on an automata. The last part identifies phases, which is time intervals in the cycles. The process in the main algorithm is shown in figure 3.3.

Figure 3.3 shows the identification order; events, cycles and phases. Events are identified in the dataset by identifying distinct operations. Temporal reasoning is made to identify an event. When all the events are identified they are sorted after appearance and predefined cycles are identified by an automata. To increase the identification of cycles a probability function is implemented. The probability

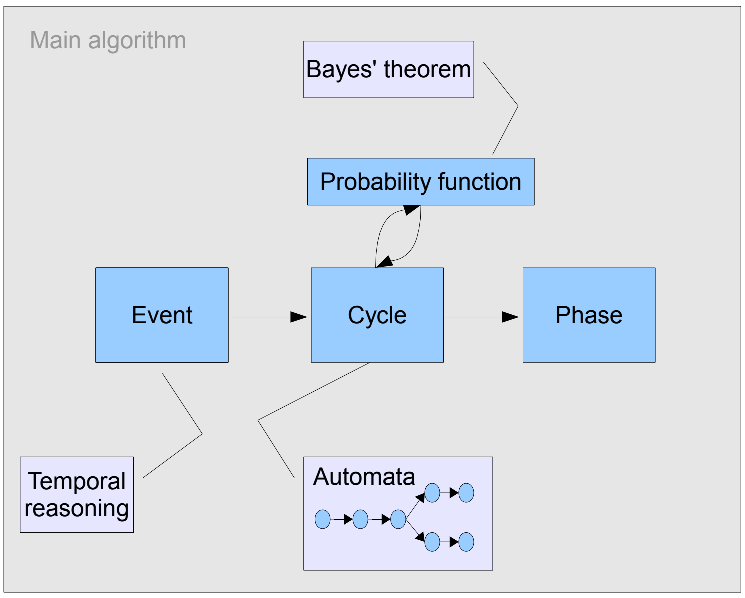


Figure 3.3: A schematic view over the main algorithm, with the identifications order. The boxes that are connected with a bent line are applied techniques that are explained in chapter 2.

function also make the algorithm more robust because it identifies cycles that are missing an event. After the cycles have been identified they are divided into phases. The division is made by using both events and cycles. The phase identification is made in the end of the main algorithm because it makes it more robust.

3.2.1 Events

At first the signals are compared with models for different typical operation events. Examples of that are driving forward or reverse. In this way it is possible to categorize the dataset in operated events. The nine events that the algorithm identifies are tabulated in table 3.2. In the table loading events and unloading events are made as events for three different attachments. The events operates differently according to the attachment but they have the same purpose. Having different events depending on the attachment makes it possible to separate the handling by attachment. To separate bucket and pallet attachment a starting condition in the event identification is made on the mean tilt angle for a time sequence. If the tilt function is used a lot then the probability is high that it is bucket handling, and if the variance of the tilt angle is small then it is high probability for pallet handling. Timber handling is separated from the other handling by the use of the third function lever. The third function makes the claw to open or close. All events are given a sample index and are sorted by this after of the event identification.

Table 3.2: Table over events that the algorithm identifies.

Event	**Abbreviation**	**Description**
Forward	f	Vehicle direction changes to a forward motion.
Reverse	r	Motion change to backwards.
Bucket loading	l	The bucket is loaded.
Bucket unloading	u	The bucket is emptied.
Pallet loading	l	The forks are loaded.
Pallet unloading	u	The forks are unloaded.
Timber loading	l	The claw is loaded.
Timber unloading	u	The claw is emptied.
Standing still	s	Standing totally still for over 1 minute.

For an event to be identified a few signals need to fulfill the conditions that are specified for a certain event. Below the conditions for the different events are described.

Speed events

There are two types of speed events in the algorithm, driving forward, *f*, and driving in reverse, *r*. To identify speed events the vehicle velocity is calculated from the drive shaft angular speed $|\omega_{ds}|$ and the direction of the gear lever to decide the driving direction. For f to be generated an interval should exist where the vehicles velocity, v is negative at the start and positive in the end of the interval. The interval between the start and end might be zero. This can be described by;

$$v_j < 0 \; and \; v_{j+1} = \ldots = v_{k-1} = 0 \; and \; v_k > 0 \tag{3.2}$$

Note that length of the interval is not fixed but depends of the numbers of 0 velocity in between the start and end of the interval. For reverse driving there is a corresponding condition;

$$v_j > 0 \; and \; v_{j+1} = \ldots = v_{k-1} = 0 \; and \; v_k < 0 \tag{3.3}$$

Bucket events

The bucket events contain loading and unloading of the bucket. Conditions are made on the lift and tilt angle and also on the direction of the gear lever. To make it possible to identify all events, are the interesting signals for bucket studied in time windows and the conditions are tested. The time windows moves forward with one sample at the time. The size of the section or time window is 4 seconds, which is a design parameter that was tested, how many samples are included depends on the sampling frequency. When the conditions for loading are fulfilled the start time is corrected, that means that the derivative of the lift angle is studied backwards in time to find the point where the lift arm does not increase any more and this point is marked as the starting point of the load event. If the loading sequence is longer then 4 seconds the event can be detected more then once, it detects every time the conditions are fulfilled. But to only identify one event for every loading sequence, only the first detected event with corrected start time is identified as the loading event. The time window strides forward with one sample at the time which makes the event identifications independent of the length of the actual loading sequence.

Algorithm Pseudo code for the bucket event identification.

```
for $i = 1 : \mathrm{N}$
  comment:  N is the length of the dataset
  if $mean\,\theta > \kappa_{mean\theta}$
    comment: : $\theta$ is the lift angle. $\kappa$ are parameters.
    if $\Delta\phi > \kappa_\phi \,\&\, \Delta\theta > \kappa_{\theta_{load}} \,\&\, \theta(i) > \kappa_{\theta(i)} \,\&\, gear\,direction\; = \; forward$
      comment: : A load event has happened.
      $Correct\,start\,time\,and\,store\,load\,event$
  else if $\theta(i) < \kappa_{\theta(i)} \,\&\, \theta(i-1) \geq \kappa_{\theta(i-1)}$
        comment:  Unloading event has happened.
        $Store\,unload\,event$
      end
    end
  end
```

When a loading event is detected the algorithm first checks if a loading event is identified in the earlier time window. An event is identified if the time range between this identification and the earlier one is smaller then the time used in this time window and the time in the earlier one. If that is the case this event is connected to the earlier and the start time is not stored. Else if the detected load event is the first the function looks for the actual start position, that means when the driver accelerates and starts driving into the pile. This is made stepwise backwards by studying the derivative of the lift angle (θ). When the movement of the lift angle is small the nearest (backward in time) sample position is set to the start of the loading event.

In the pseudo code it is possible to see the conditions on the signals, κ are parameters for the detection. The conditions for the loading event is that the lift arm is raising and in the same time the bucket is tilted up, the gear lever direction should also be forward. The conditions for the unloading event is the conditions on the tilt angle, because it is possible to unload in many different heights which makes a specific requirement on the lift angle unnecessary. The condition that the algorithm uses to identify bucket unloading, is when the tilt angle has reached the point where the bucket is emptying everything it holds.

Pallet events

The idea for the detection of a pallet loading event is to identify the positioning of the wheel loader before the loading process. For experienced drivers the time for positioning can be very short. The function for identifying pallet events has been made in two versions, the first one search in an interval that is chosen between the times the gear lever is put in reverse after ls pressure (see figure 1.1) over the average. This function was not working exactly as hoped and when it was tested inside the rest of the algorithm it blocked the possibility to identify bucket unload events. Therefore another pallet event identification function was developed, which do not use the ls pressure in the same extension because the pressure can be affected by more then the lift arm. At first all the positions where the vehicle direction is changed to reverse are detected. Between two of these points, if the number of samples is larger than two, the positioning is searched for. The first thing that is studied is the vehicle speed. To position the wheel loader the vehicle speed, v, should be small. A loading or unloading event is identified when the derivative of the filtrated lift angle, $\Delta\theta$, is smaller than a design parameter, κ_i, i.e.

$$\Delta\theta < \kappa_i \tag{3.4}$$

The mean output torque should also have a value between two design parameters, t_i and t_j, $t_i < mean(output\ torque) < t_j$ and the ls pressure should be greater than a parameter, l_i. When all these conditions are fulfilled a pallet load/unload event is identified.

Timber events

When the handling material is timber, the third and fourth functions[1] are used. The third and fourth function are not specific for the timber claw, but in this thesis this is the only attachment that uses the third and fourth function. A positive result on the lever for the third function makes the claw open and a negative result makes the claw close. To identify a timber event the signals for third lever position and vehicle speed are used. First all moments when the third lever position is not equal to zero are picked out. Then the algorithm iterates over the found positions, for every position the lever is checked if it has a positive or a negative result. The speed is also monitored, and if it is near or equal to zero and in the same time the third function lever is separated from zero, a load or unload event has been identified.

If the lever position for the third function is greater than zero the claw is closing and indicates that a loading event is happening. The same applies when the third function signal is less than zero for an unloading event. Pseudo code for the function for identifying timber event:

[1]The third and fourth function are hydraulic connection for manoeuvre some attachment. The third function controls the opening and closing of the claw. The fourth function controls a push inside the claw that pushes together the timber.

Algorithm Pseudo code for the pallet event identification.

```
$point = find(Third\ lever! = 0)$
for i=point(start):point(end)
  if $ThirdLever(i) > 0\, and\, speed(i) == 0$
    comment: The third lever is greater than zero and
    comment: there is no velocity => assuming loading.
    if i-temp_load > 2
      comment: If more than two samples between previous
      comment: stored load event and detected events, store the event.
      $Store\, load\, event$
    end
    $temp\_load = i$
 else if $ThirdLever(i) > 0$
      comment: Third lever is greater then zero, can be loading.
      $temp\_load = i$
  else if $ThirdLever(i) < 0\, and\, speed(i) == 0$
        comment: Third lever is smaller than zero and
        comment: no velocity => assuming unloading.
        if i-temp_unload >2
          comment: More then two samples between stored and this unload event.
          $Store\, unload\, event$
        end
        $temp\_unload = i$
    else
        $temp\_unload = i$
      end
    end
  end
end
end
```

Standing still events

The event for standing still has more criteria than most of the other events because it needs that many signals are zero at the same time. This event is identified because the time when the wheel loader is not used can be removed and not be treated when characteristic parameters are calculated. To say that the wheel loader is standing still, needs that the gear lever is put in neutral, no movement on lift and tilt lever and also that the accelerator is zero. When all of this is fulfilled for more than 15 seconds, the algorithm identifies that the vehicle is standing still. The signals for lift, tilt, gear and accelerator are observed because when they are zero the machine really stands still and does not make any movement at all. Below are the code for identifying standing still event.

The code for identification of standing still

```
for i = 1+W:W:N
    speed_v = VehicleSpe(i-W:i);
    gearDi = mean(ActualGearDi(i-W:i));
    acc = var(round(AcceleratorP(i-W:i)));
    lift = var(FirstLevPos(i-W:i));
    tilt = var(SecLevPos(i-W:i));
    m_g = round(mean(TrueVehiDi));
    m = mean(abs(speed_v));
    v = var(speed_v);
    n = find(speed_v <= 0);
    if lift == 0 && tilt == 0 && v == 0 && m_g == 0
         pos = i-(W-n);
         ss_pos = [ss_pos pos(1)]; %Store standing still position
    elseif gearDi == 0 && acc == 0 && lift == 0 && tilt == 0
         pos = i-(W-n);
         ss_pos = [ss_pos pos(1)]; %Store standing still position
    end
end
```

Where W is the time window and N the length of the dataset. The size of the time window for standing still event is 15 seconds[2]. The time window steps forward through the dataset with one step at the time. The start time of the time window is set to the identification index when an event has been detected. The standing still event is used to get the start time from the time when algorithm identifies the event. The speed events are only identified when a transition is made on the gear lever, in contrast to them the standing still event can be identified multiple times after each other.

3.2.2 Cycles

When all events have been detected they are ordered chronologically. To identify a cycle, an automata is defined for the cycle, in figure 3.4 the automata for bucket main cycle is shown.
The symbol f stands for forward driving, l loading, r reverse driving and u for unload. To make the cycle identification more robust a probability function is connected to the cycle identification. The probability function is described in section 3.3.

Cycle identification for main cycles have one automata for every attachment, (bucket, pallet or timber claw) but the structure is the same. The automata for pallet handling is illustrated in figure 3.5.

[2]The size of the time window is a design parameter and is chosen that big since if the vehicle stands still it happens often for minutes.

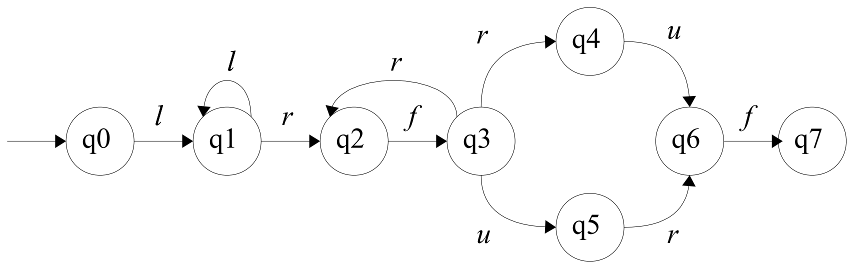


Figure 3.4: Transition diagram over automata for loading cycle, in this case bucket loading.

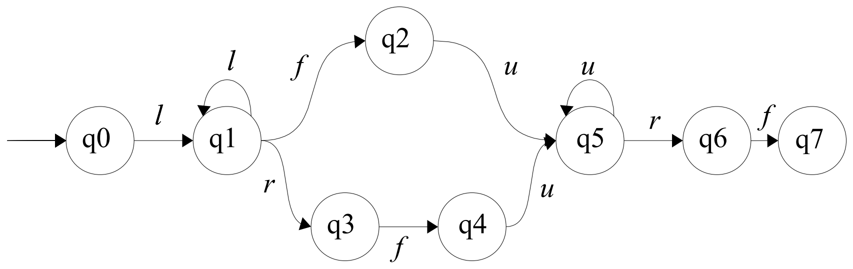


Figure 3.5: Transition diagram over pallet cycle.

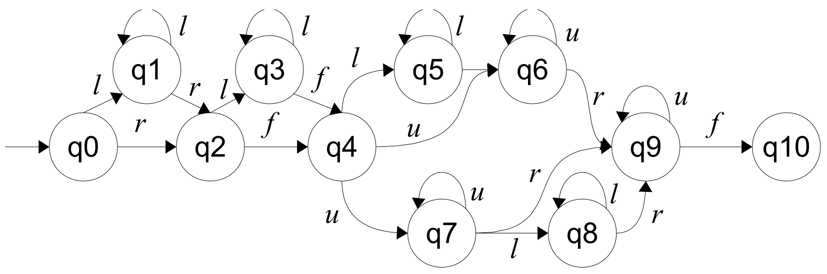


Figure 3.6: Transition diagram over timber cycle.

Figure 3.6 illustrates the cycle pattern for timber handling. The pattern is so complex because it compensate for the not so good event identification. Note that more than one type of cycle can be identified for a dataset and different attachments can be used. When the detection for main cycles are completed, the algorithm starts to search for sub cycles (section 1.1.2). The automata used for sub cycles can be seen in figure 3.7.

The algorithm manages to identify sub cycles even if no main cycles have been

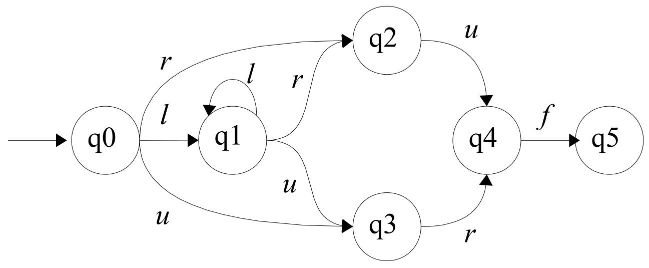


Figure 3.7: Transition diagram for a cleaning cycle automata.

identified. Note that the sub cycle is only defined for bucket handling because cleaning is used there.

3.2.3 Phases

A phase is a time interval and a distinct part of the cycle. To make the phases identification robust the phase division is made after the cycle identification. The algorithm uses both the identified cycles and events to split the cycles into four phases;

- Loading (noted α)
- Drive loaded (noted β)
- Unload (noted γ)
- Drive unloaded (noted δ)

Phases are parts of the cycle which are needed to calculate characteristic parameters for suitable parts of the cycle to get the most resembled information about the usage. A characteristic parameter can be the speed in the driving phases and the lift speed for the loading and unloading phase, all parameters are tabulated in appendix A.1.

The phase part of the algorithm starts with selecting the start point for all identified cycles. A phase starts when the event that is characteristic for the phase starts, for loading it is when the loading event starts and for driving loaded or unloaded it is when the vehicle direction changes. The phase separation works on main cycles and cleaning cycles. In cleaning cycles the phases are transport and cleaning.

3.3 Probability function

The purpose of the probability function is to increase the robustness of identifying cycles when an event is not identified (i.e. different manoeuvres than the specified event conditions) or when small changes are made in the cycle pattern. The cycles that are identified when one event is missing or identified in the reverse order is called a corrupted cycle. The probability function is connected to the cycle identification function. The function sends information about number of identified cycles, the pattern for the cycles and the probability. Note that it is possible to send the information that no cycles have been identified. The probability for a pattern is calculated from how many cycles that have appeared in the dataset. The frequency of handled cycles is subtracted from the highest probability. The frequency of handled cycles is small when many cycles have been identified, which gives a high probability for that type of cycle and the other way around for a few identified cycles. The working process can be described as:

1. Find standing still events or sequence of events and calculate the total standing still time.
2. Search for corrupt cycles.
3. Make sure that the identified corrupt cycles are not a part of correct cycles.
4. If so, remove them from the corrupt cycles.
5. Calculate the mean cycle time, for the corrupt cycles.
6. Remove the time for standing still from the total time. Calculate how many corrupt cycles, N, with the mean cycle time that can appear in the remaining total time.
7. Calculate the probability for the corrupt cycles according to: (number of corrupt cycles)/ N.
8. Use Bayes' theorem to decide if the corrupt cycles are probable.

The first thing that is made in the probability function is to identify all standing still events (section 3.2.1) and calculate the time that the wheel loader is standing still. The calculated time is subtracted from the total datasets time to get the real time that the wheel loader has been operated.

The events are run through an automata that represent corrupted cycles. The automata for corrupt cycles has been developed from cycles that the algorithm did not identify and can be seen in figure 3.8.

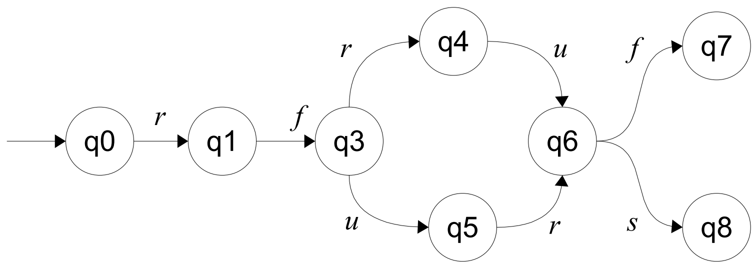


Figure 3.8: Transition diagram for a corrupted cycle automata.

If corrupted cycles are identified the end index is compared with the end index for earlier identified main cycles, so that cycles that are identified twice can be removed from the corrupted cycles. Then the corrupted cycles are calculated and also the probability for them. The probability is calculated from the mean time for the cycle divided by the possible number of cycles for the time in the dataset when the time that the wheel loader is standing still is subtracted. To decide if the corrupted cycles are probable, Bayes' theorem (equation 2.3) is used. If the conditional probability is higher than a design parameter the corrupted cycles are included to the detected main cycles. The design parameter is chosen to 60% which gives a high probability for the identified corrupt cycles to be probable.

The only difference that is made if no main cycles have appeared is that the likelihood for earlier appeared cycles are given the reverse probability for the corrupted cycle.

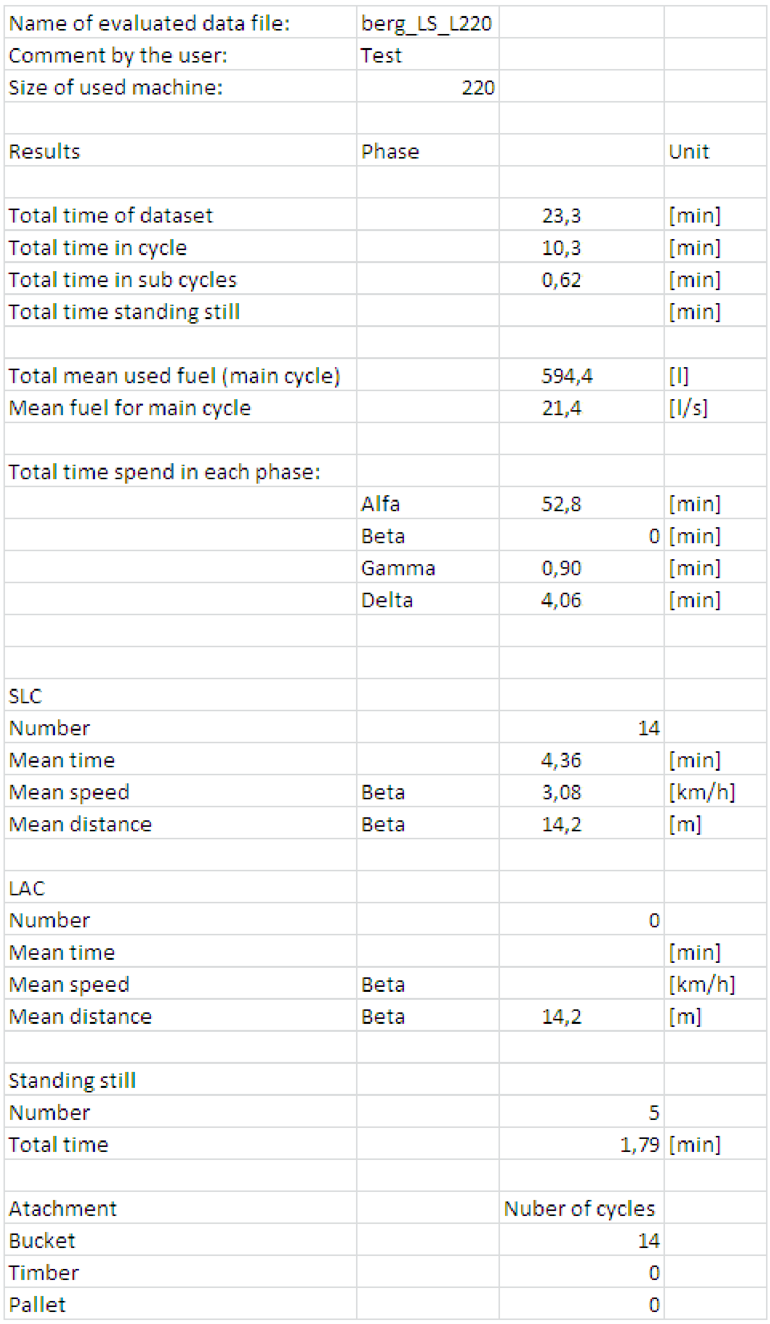
3.4 Post algorithm

When the main algorithm has detected all cyclic behavior in the operation data, the information is passed to the post algorithm. Characteristic parameters are calculated per cycle and per phase in the post algorithm. Which parameters that are calculated and for which phase are tabulated in table A.1.

After all parameters have been calculated the user is given the question if they want to see plots (see user guide in appendix C). Plots that are displaced are listed in table B.1.

When all the plots have been displayed the user get another question, if they want to store the parameters in an excel file. If the answer on that question is yes is an excel file created with the same name as the dataset that is evaluated. The excel file contains two sheets. Information about the dataset like name of the dataset, comments and size of the vehicle are displayed first on the first sheet. Then follows information about time, dataset, in cycles, in sub cycles and standing still and for the main cycles total time that has spent in every phase. In figure 3.9 the first sheet is displayed.

In the second sheet are parameters for all cycles presented per phase. The algorithm ends after the excel file has been produced. The last thing that is made before the end is to delete the temporary file with the processed signals. If the user does not choose to store the parameters in an excel file the algorithm ends like the previous case and the parameters can be read in *results* in the workspace.



Name of evaluated data file:	berg_LS_L220		
Comment by the user:	Test		
Size of used machine:	220		
Results	Phase		Unit
Total time of dataset		23,3	[min]
Total time in cycle		10,3	[min]
Total time in sub cycles		0,62	[min]
Total time standing still			[min]
Total mean used fuel (main cycle)		594,4	[l]
Mean fuel for main cycle		21,4	[l/s]
Total time spend in each phase:			
	Alfa	52,8	[min]
	Beta	0	[min]
	Gamma	0,90	[min]
	Delta	4,06	[min]
SLC			
Number		14	
Mean time		4,36	[min]
Mean speed	Beta	3,08	[km/h]
Mean distance	Beta	14,2	[m]
LAC			
Number		0	
Mean time			[min]
Mean speed	Beta		[km/h]
Mean distance	Beta	14,2	[m]
Standing still			
Number		5	
Total time		1,79	[min]
Atachment		Nuber of cycles	
Bucket		14	
Timber		0	
Pallet		0	

Figure 3.9: Example of first parameter sheet that can be stored. The algorithm wrights the parameters calculated value without rounding.

Chapter 4

Results

Experimental results from the master thesis are presented in this chapter. The outcome of the algorithm is separated by handling area.

4.1 Bucket handling

Most of the provided data were bucket handling with different types of cycles and handling materials. Different cycles and handling materials gives large variations in operating conditions which is good for evaluating the algorithms. To validate bucket handling, data with three different types of main occurrence handling cycles are used, tabulated in table 4.1. The handling cycles are; SLC loading gravel, LAC loading gravel and SLC loading rocks that imitate shoot rocks, the last one is a tricky material to handle and the loading part can take long time.

Table 4.1: Table over used dataset.

Dataset number	Description
1	SLC gravel
2	LAC gravel
3	SLC rock[1]

[1]Rocks that should imitate shoot rocks, hard to load.

Each one of the handling areas in table 4.1 were operated by three different operators. The operators have different experience levels, one is a beginner, one is intermediate and one is an experienced driver. This gives a wide range of vehicle handling to validate the robustness of the algorithm. The operators are given a number, tabulated in table 4.2.

Table 4.2: Table over the operator used for the validation.

Number	Description
1	Experienced
2	Intermediate
3	Beginner

To summarize, nine datasets are used to validate the algorithm. To confirm in a secure way if the number of identified cycles is correct, the cycles are manually identified from the front window movie. The movie made it possible to validate if the algorithm has correctly identified the cycles. In the center column in table 4.3 the number of cycles identified from the movie are presented. The first two columns contain information about which dataset and operator that are validated. Next to the column with the manual number of cycles is a column with the number of cycles that the algorithm has identified. A percental number is presented in the last column which tells how well the algorithm identifies the pre defined cycles.

Table 4.3: Table over cycles that the algorithm identifies.

Data	Operator	Actual number of cycles	Cycles detected	Per cent
1	1	15	15	100
1	2	15	15	100
1	3	12	9	75
2	1	13	13	100
2	2	15	15	100
2	3	6	6	100
3	1	15	14	93,3
3	2	18	18	100
3	3	10	9	90

The percental numbers are high, over 90 %, for all but one that is slightly lower in the last column in table 4.3. This dataset has a lower identification per cent because the operator operates far from the pattern of the pre defined cycles. One of the three undetected cycles is not identified because the operator does not switch the gear lever as; reverse to neutral and than stand still. Instead the operator is doing the following maneuver; reverse to neutral to reverse to neutral and then stand still. For the other validations it can be read from the table that the algorithm has the best result for LAC, which identified 100 % irrespectively of the experience level of the operator. Besides good performance on LAC the algorithm identifies at least 90 % of the cycles when rocks, that imitate shoot rocks, are the handling material. The conclusion for bucket handling is that the algorithm performs well independent on the cycle, handling material and the experience level of the operator.

In figure 4.1 all identified load and unload events are displayed as stars on the lift and tilt signals. The most interesting in this figure are the loading events which are identified when the lift arm (continuous blue line) increases at the same time as the tilt angle increases. The loading events are identified when loading are being done, which can be seen in the figure. The unloading events are identified when the tilt angles (dashed green line) decreases. It can be seen in the figure that the algorithm has identified one unloading event every time the tilt angle decreases.

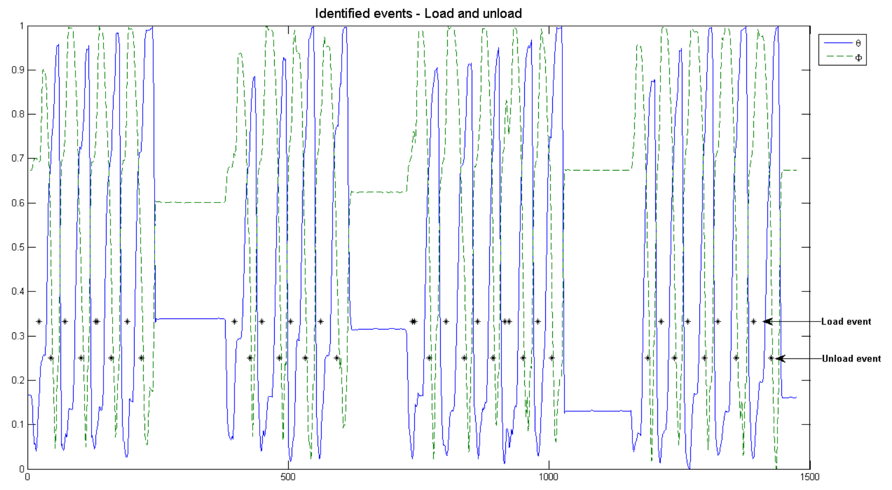


Figure 4.1: Identified load and unload events.

Figure 4.2 illustrates driving forward, reverse and standing still events for the same dataset as the load and unload events. The signal is the calculated vehicle velocity that is positive when the wheel loader is travelling forward and negative when travelling in reverse. The algorithm should identify one driving forward event every time the velocity becomes positive, which is illustrated with a star at level four in the figure. In the same way should the algorithm identify a reverse driving event every time the velocity switches to negative, illustrated with a star at level 2 in figure 4.2. The identified standing still events are illustrated with a star at the zero level in the figure, which tells that the vehicle velocity is zero. To summarize is the events driving forward, reverse and standing still identified on the correct place.

The used dataset is number 3 (table 4.1) and the experience level of the operator is 2 (table 4.2). Where the algorithm identifies the cycles are shown in figure 4.3. The signals are lift angle (continuous blue line) and tilt angle (dashed green line). The identified cycles are illustrated with a shadowed area, in the white area no

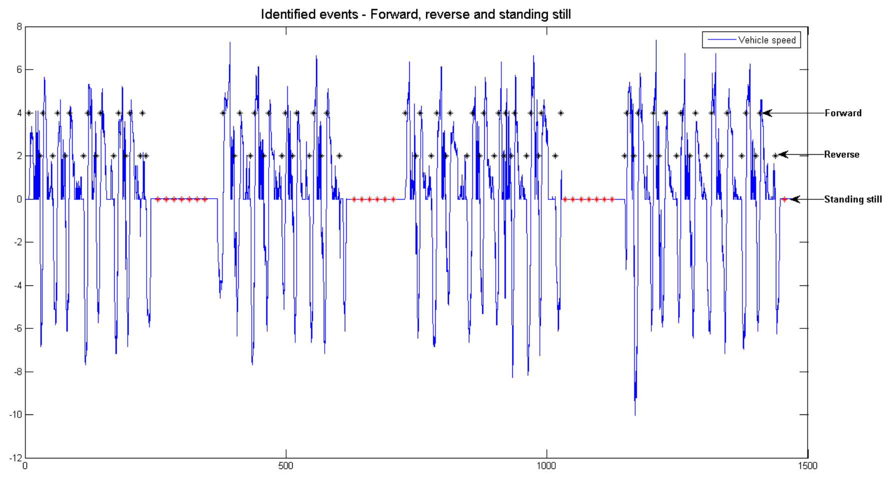


Figure 4.2: Events connected to motions as driving forward or in reverse and standing still.

cycles are identified, it possible to see that both lift and tilt are not much used in the white areas. A cycle starts at the point where the loading event is identified, figure 4.1, and ends after a driving forward event, 4.2. It can be seen in the figure that all the cycles are identified which also is the result in table 4.3.

4.2 Pallet handling

The number of datasets with pallet handling was not as many as with bucket handling. For pallet handling, the signal for the vehicle direction was missing for all datasets. The signal was recreated by looking at the movie from the front window and mark when a direction change occurred. For the dataset that this signal was made from, the tilt and lift signal were also corrupt but small pieces were correct. That these signals were corrupt was detected after the vehicle direction signals were recreated. The time to recreate a new vehicle direction would be long and the remaining time of the thesis were short so therefore were the pallet identification only tried at the applicable parts of the pallet datasets. Figure 4.4 shows the lift and tilt angle together with the load and unload events. Comparing this graph with 4.3 repetitive behaviour on lift and tilt signals is hard to find in 4.4. A repetitive behaviour can be seen by watching the pallet sequence in the movie.

In table 4.4 is the outcome of the algorithm, in the second column are the times when the event should appear and in the third where the algorithm identifies them or the closest ones in column four. When looking in figure 4.4 some of the iden-

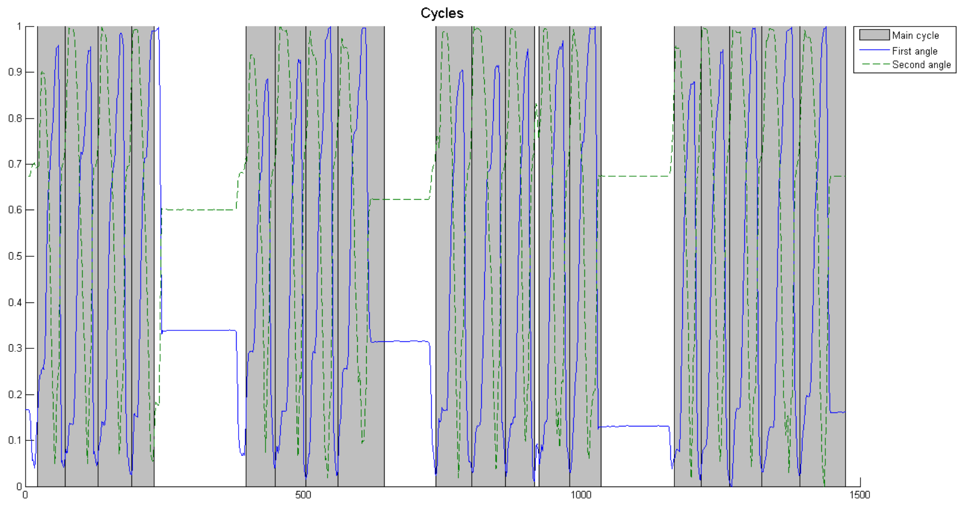


Figure 4.3: Identified cycles for dataset 3 for driver 2.

Table 4.4: Table over where the events should be identified and where the function identifies them, connected to figure 4.4. The last column contains events that are detected closest to the actual events.

Event	Movie	Algorithm	Closest detected event
Load	9075,24		9065 or 9087
Unload	9108,25	9109	
Load	9126,24		
Unload	9141,24	9140	
Load	9160,25		9169
Unload	9191,24		
Load	9223,24		9218
Unload	9256,25	9258	
Load	9344,24		9339
Load	9376,25		
Unload	9409,24	9400	

tified pallet events are marked with the time and it is hard to see if loading or unloading is done only by studying the signals.

To validate the developed pallet events function it was tested on another part of the dataset. The function did not work, it did not identify pallet events even near where it should have. The pallet function was implemented in the algorithm but then it blocked the algorithm to identify bucket events. The event function for

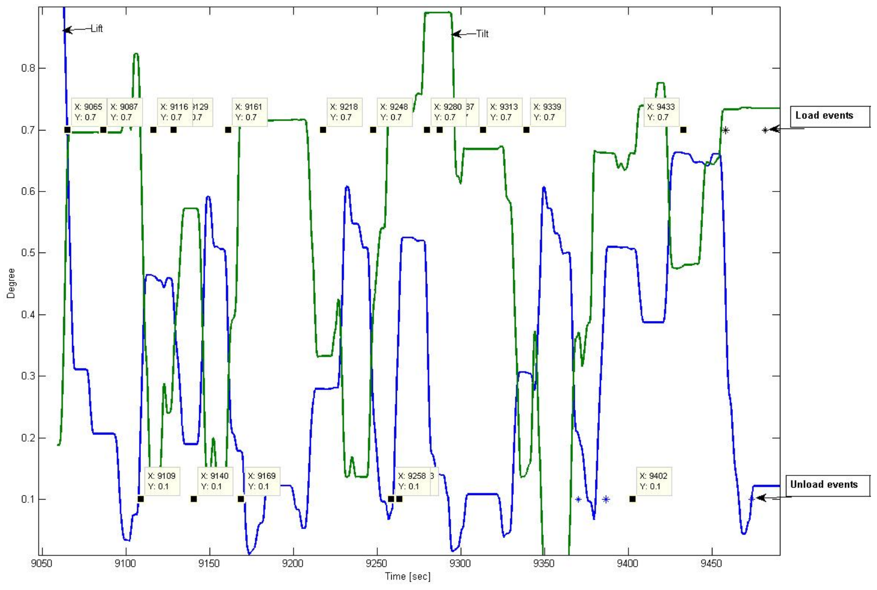


Figure 4.4: Angles normalized and the detected events. The squares are marked start, the box with the coordinates belongs to the square.

pallet handling was then remade to get a better identification and not block bucket handling. The result from remade pallet identification is displayed in figure 4.5. This second function only detects when a load or unload event has happened, it can not separate them apart.

Which events that are identified at the right place are tabulated in table 4.5. There are both the correct time for the events and where the function identified them.

This second function is implemented in the algorithm but it have only been tested on one other dataset. The result for that is discussed in section 4.4.

The conclusion for the pallet handling is that a much more extensive temporal reasoning is needed to detect pallet handling to be capable of separating loading and unloading operations. It is hard to separate loading and unloading because this happens on different heights on the lift arm and the weight on the load was small. Loading height in one cycle can in the next be the unloading height, this causes problem. To identify pallet events the positioning before the actual event needs to be identified, for the handed over datasets are this positioning so small that it is almost impossible to see.

Table 4.5: Table over pallet events and where the function identifies them, connected to figure 4.5.

Event	Movie	Algorithm	Closest detected event
Load	15130.2		15150
Unload	15199.2	15210	
Load	15394.2		15370
Unload	15456.2		15480
Load	15552.2		15500
Unload	15654.2		15600
Load	15792.2		15710
Unload	15861.2	15860	
Load	15902.2		
Unload	15922.2		15950
Load	15939.2		15970
Unload	15980.2		16000
Load	16059.2		16000
Unload	16142.2	16140	

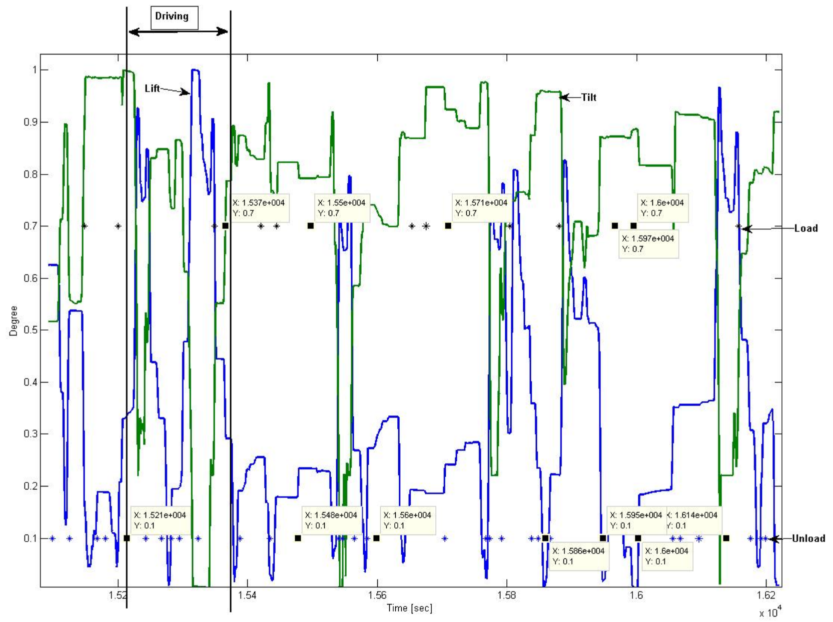


Figure 4.5: Angles normalized and the detected events for function number two.

4.3 Timber handling

In the first dataset for timber handling, the signals for the third and the fourth function were missing. This made the data unusable for developing timber events because the third function is connected to open and close the claw, which is central in the timber handling.

In the later part of the thesis a new measurement series was made. A draft for an event identification function was developed. By using the third function together with the vehicle speed, timber events can be identified. In figure 4.6 shows the identified cycles.

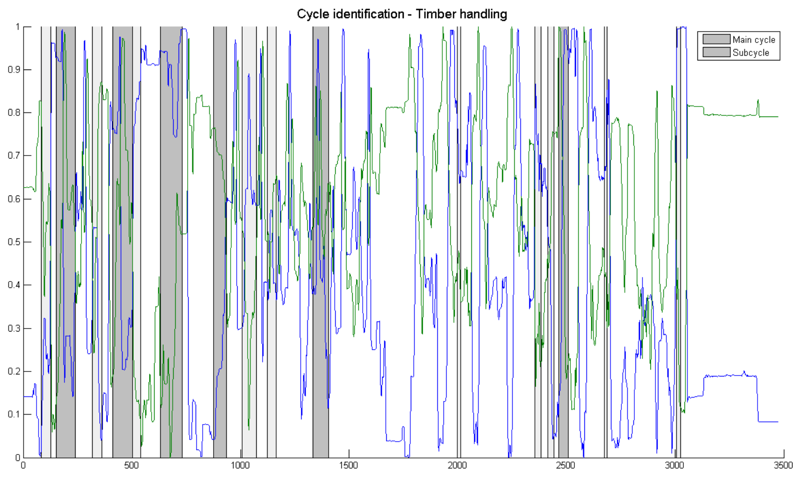


Figure 4.6: Shadowed areas are identified cycles. The signals are tilt and lift angle normalized.

In figure 4.6 it is clear that the algorithm is not good on identifying timber cycles. Figure 4.7 gives a view over how much time that is identified as cycle treatment. The main reason that the algorithm did not identify more cycles correctly is due to the fact that the timber event function identifies too many events. The reason for this is that the third function lever is used more than just at the loading and unloading part in the cycles. If the timber event function is developed to be more robust and identifies the loading and unloading event at the right place the algorithm should identify more cycles.

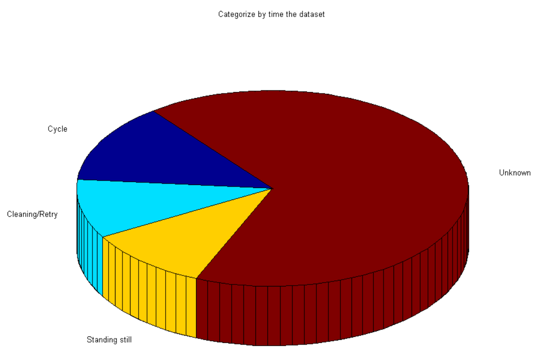


Figure 4.7: Pie chart over time distribution according to the algorithm.

4.4 Mixed cycles and attachment

A mixed dataset, containing two different attachments, bucket and pallet were tested. A selection of signals that show repetitive behavior are showed in figure 4.8. Two different kinds of cycles are operated; it starts with LAC with a bucket as attachment followed by a few cleaning cycles and a transport phase. Then enters the SLC, still with the bucket this is followed by a few cleaning cycles and a transport phase. After this transport phase, the attachment is switched to pallet and then SLC is made with the pallets. In the last section in figure 4.8 it is hard to see any repetitive behavior.

Which cycles the algorithm identified are seen in figure 4.9. The algorithm identified all cycles in the part with SLC for bucket handling. The algorithm only missed one cycle (taken for a sub cycle) in the LAC part. In the part with the pallet attachment, only a few cycles are marked, because of the lack of repetitive behavior and the problem with identifying pallet load and unload events.

The results for the bucket part of the mixed dataset are listed in table 4.6.

Table 4.6: Table over cycles that the algorithm identifies, for the bucket part of the mixed dataset.

Cycle	Actual number of cycles	Cycles detected	Per cent
LAC, bucket	10	9	90
SLC, bucket	17	17	100

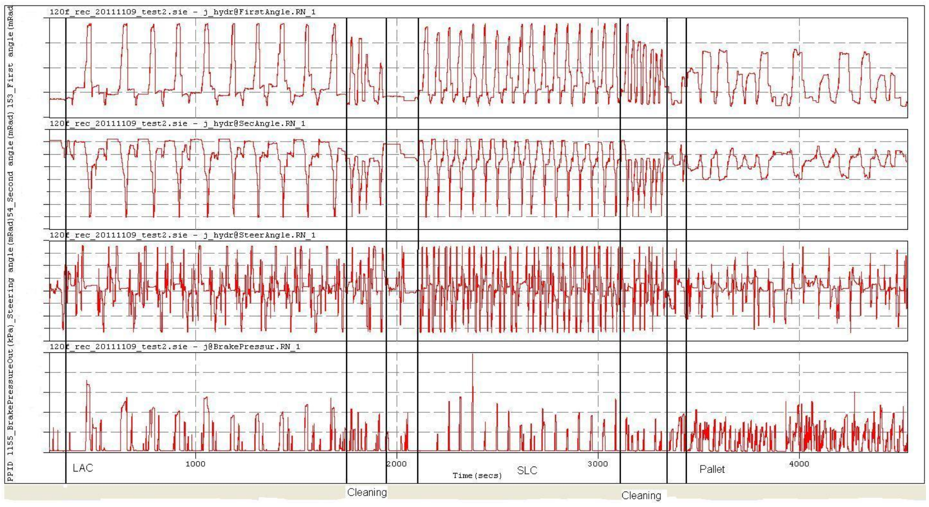


Figure 4.8: Section of the operated cycles. The signals are; lift angle, tilt angle, steering angle and brake pressure. Non repetitive behavior are seen in the last sequence.

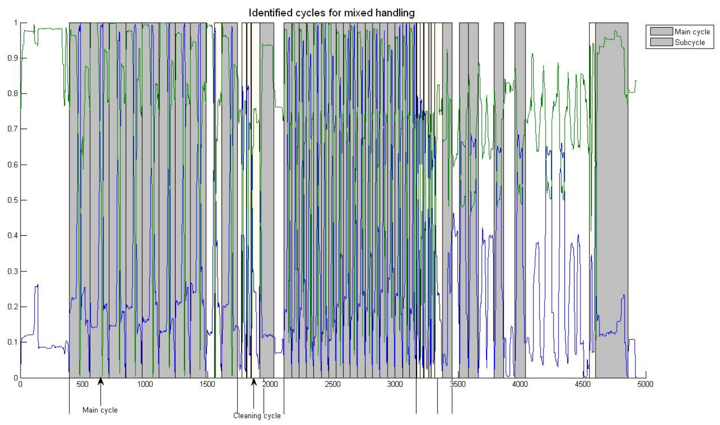


Figure 4.9: Shadowed areas illustrate identified cycles by the algorithm. The lines under the plot are the same markers as for figure 4.8.

Chapter 5

Discussion

This chapter contains discussions of different design decisions for the algorithm. Methods that have been attempted during the thesis in different areas are presented and discussed.

5.1 Upsampling

To get all the signals at the same sampling frequency, upsampling was needed, more than one method have been evaluated. The idea behind this design choice was that the algorithm should be independent of the sampling frequency with which the signals where sampled from the CAN bus (controller area network). The sampling frequency affects identification of events, because they are studied in a time window. How fast or slow the sampling frequency is, affects how many samples that are studied, to decide if an event has appeared or not. Instead, if all signals have the same sampling frequency, the size of the time window can be adaptive and decided based on the samplings frequency

$$T = \frac{1}{f} \tag{5.1}$$

where f is the frequency. To be able to do this, the algorithm needs the information about the current sampling frequency. This information is collected by using the length of a signal with the highest frequency and divide this with the length of a signal with the lowest sampling frequency, which is assumed to be 5 Hz, because the lowest sample frequency for all handled datasets were 5 Hz. The factor that the deviation gives, multiplied by 5 to get the highest sampling frequency.

Another method that was first attempted was zero padding. The implemented zero padding was using fast Fourier transform and two different cases. At first the cases were even or odd signals that needed upsampling. After a while, a new problem was arising. More cases were needed if the outgoing signal should be even or odd, if the incoming signal was of the opposite type. The final length of the signal was also causing a problem. Quickly there were many different cases and special cases which resulted in different ways of zero-padding and the result was not satisfying. The handled signal was hissy and the amplitude was reduced in half. The different cases also affected the performance of the algorithm. Large quantities of code was needed to decide which of the cases should be used for zero-padding. The pre-algorithm was because of this very time consuming and made the whole algorithm very slow.

5.2 Time window

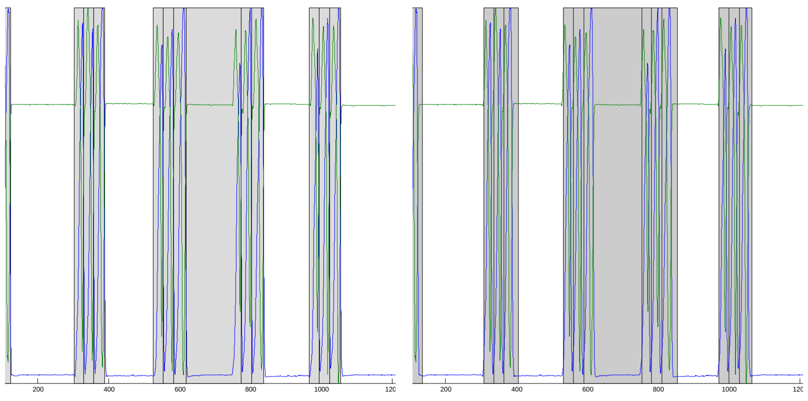
The number of samples that are needed to identify an event is limited by the sampling frequency. A number of samples may correspond to one time window for one sampling frequency and another size of the time window for another frequency. To improve this, the size of the time window for the signals and how many samples that are studied is decided from the frequency using the function 5.1. The identification of all events is sensitive to the size of the time window. Tiny changes, of the time window gave large changes on how good the algorithm was at identifying events. The times that are set in the algorithm are tested out to give the optimal event detection for the given datasets. The tests are done manually and the size of the time window is set as a constant in the algorithm. How the size of the time window is set can affect the outcome of the algorithm. One way to get around this is to implement a loop that single out the different optimal time window sizes for the current dataset. Another approach is to have adaptive time windows that adjust after certain event detections. This approach should be more robust against different cycle times.

One design choice concerning the time windows is if they overlap, i.e. the start time for the next coming window is inside the active time window. There are two ways to do this. One way is to move the time window a decided number of samples. This type of overlapping windows can cause a large number of event detections for the same event. Because of this there is a need for a function that compares the event's start time with the signal and deletes those that are included in the previous event. Another way to overlap the windows is to set the start point of the next window to the point where the event was found. If this method is used it can be easier to collect events that actually are the same event.

After studying many datasets, it seems that the best way to choose the size of the time window is to have an adaptive size of the window, where the sizes are decided from reasonable event appearances. An example of that, for bucket handling, can be to have the time window between two positive peaks of the lift lever or between two direction events, like driving forward and then reverse, where a bucket loading should be identified. Similarly, when identifying the unloading event, the time window could be between two occurrences of driving forward. For more distinct event identification, an adaptive window size is probably better. Presumably it does not cause any problems when shot rock is the material that is loaded.

5.3 Change automata order

Tests on changing the automata order were carried out. Instead of starting with driving forward and then load, the automata was started with loading and ended with driving forward. The new automata order is the one that has been used for the thesis work and is seen in chapter 3.2.1. The pattern for sub cycles was also changed in the same way. The result of the automata change was that all the cycles were identified a bit different, which can be seen in figure 5.1b. In figure 5.1a the outcome from the algorithm when the automata order was not changed is presented. From the plots it is possible to see that there is not a big difference between the outcomes of the two automata orders. The biggest difference is between the third and fourth cycle sequences.



(a) Unchanged order (b) Changed order

Figure 5.1: To the left is the outcome from the algorithm when loading starts the cycle sequence and to the right when the cycle start when the wheel loaders starts to drive forward

The dark partition in figure 5.1a and 5.1b illustrates cycles that have been identified by the algorithm. In the long white partition, the wheel loader is standing still with no gear and no movement on the levers for lift and tilt (section 3.2.1). The reason why the other sequence, that looks like the standing still sequence, is not identified as one is that the driver has put the gear lever in forward and thus the conditions for standing still are not fulfilled. The algorithm that has the unchanged automata order starts a cycle identification before the long section that looks like standing still. The algorithm with changed automata order ends after this section. The reason why they are acting in that way is that a driving forward event are identified at the start at that section and that is the start event for the unchanged and the last event for the changed automata order.

Both of the automata orders identified the cycles in the right place. The choice of which one to use is difficult. One reason to use the unchanged automata order is that it gives a distinct beginning of a cycle and if working from the assumption that the operator already has decided to operate a cycle when he or she puts the gear lever in forward. It is also easier to get a distinct end of the cycle when the automata ends when the reverse driving ends. Not much work is required to change the algorithm to work in accordance with the automata illustrated in chapter 3.2.1. The changed automata order is used because it gives a cycle identification that is more like the ones described in chapter 1.1.2.

5.4 Problem related to events

The function that identifies pallet events was difficult to design and get good detection performance. The development process has been that one dataset is analyzed to find relationships between signals that are relevant, like lift and speed, when loading and unloading pallets are made. The difficulties appears with the relationship of the signals. Such relation could not be found, there are too big differences in all of the signals. It was possible to see a relationship in only one cycle but the signals vary between the cycles. A probable reason is that the pallet handling operations varies for different operators and different sites. When the movie from the dataset was studied, one reflection was that for the most of the time a certain cycle was not operated repetitive, instead were one or two SLC made and then a big transport, a LAC or non-cycle behaviour.

The first approach was to study the derivative of lift and tilt angle combined with the ls (load sensing pump) pressure. The idea was that if an ls pressure peak with a value 4 units over the mean value occurred, a loading event happened. This seemed to work well, but the conclusion was that the signal for the ls pressure was not only affected by the lift and tilt signal, but also the steering for many machine sizes. This made that the pallet event function identified significantly more events than the ones that actually happened.

5.5 Other problems

When comparing with the video, there were sometimes interruptions in the video. This was due to the fact that the data processing software Infield cuts the signal when the engine is off but this does not happen for the video. This breaks synchronization between the signals and the video. Since the signals and the film were unsynchronized it was hard get a good view of how the signals behaved when a particular operation was performed in the video.

5.5.1 Missing actual gear direction signal

For a few of the investigated datasets, the signal for the gear lever direction was missing. More exactly the signal was constant because nothing was recorded. This impacts the information about the vehicle direction, which makes it impossible to identify speed events (3.2.1) . The first idea to go round this was to make an algorithm that calculated a direction signal. In that way the whole algorithm would be able to automatically handle the missing information. This algorithm used the signals for gear and vehicle speed and worked like:

Algorithm Pseudo code for automatic creating vehicle direction.

```
FOR i = 1 to length of signals
    IF Gear(i) = 0 and Speed(i) = 0
       direction = neutral
       flag = neutral
    ELSEIF i > 5
       IF One Speed(i-5:i) = 0 and flag != 0
          direction(i-5:i) = 0
          flag = neutral
       ELSEIF Speed(i) = 0  and Speed(i+1) > 0.1 and Gear != 0
          IF flag = positive
             flag = negative
          ELSEIF flag = negative
             flag = positive
          ELSEIF flag = neutral
             flag = positive
          END
          direction(i) = flag
       ELSE
         direction(i) = flag
       END
    ELSEIF Speed(i) = 0 and Speed(i+1) != 0 and Gear(i) != 0
       direction(i) = flag
       IF flag = positive
          flag = negative
       ELSEIF flag = negative
          flag = positive
       ELSEIF flag = neutral
          flag = positive
       END
    ELSEIF Speed(i)> 2 and flag = neutral
       direction(i) = positive
       flag = positive
    ELSE
      direction(i) = flag
    END
 END
```

The algorithm is built on assumptions and assumes that the first movement direction is forward. It also assumes that the direction is forward when the wheel loader starts to move from a standing still event. A condition that the algorithm is using is that if the vehicle speed is zero for five samples (ex f = 10 Hz => 5 samples <=> 0.5 sec) the direction is changed. This condition resulted in that the function only worked for short signals or not at all. After analysing the existing signals and trying to get a clue about how the direction signal in some cases was calculated, the conclusion was that there was no other way then to make the signal by hand. This was done by studying the video to get the time when the direction was changed, together with studying the gear signal to get the sequences where the wheel loader had a neutral gear which is assumed to correspond to standing still.

5.5.2 High or low lift

One way to decide if the lift was high or low is to use the measured value of lift and tilt angle and according to product specification calculate how high from the ground the attachment is. But this calculation has a large uncertainty, because the attachments are attached on different levels. An example: *A gravel has 490 mm PTG(Pin to ground) and a pallet have -8 mm PTG on the same size of wheel loader.* This is a big difference and if this should work in the algorithm, it needs the information about which attachment that is used and the algorithm is no longer independent on the attachment.

Chapter 6

Future work

A suggestion for future work is to adapt the algorithm to work online in the machine while it operates. A control algorithm that works together with the pattern recognition algorithm on some probability conditions is another interesting development possibility. Potentially this is a useful tool to increase the efficiency of the wheel loader, both for fuel efficiency and in productivity.

For the existing algorithm, the recognition of pallet and timber events needs to be improved. The conditions that separates the different handling operation can be improved when different attachments are used in the same dataset. Another possibility is to expand the number of supported attachments and the conditions to tell them apart. Considering that more than timber claw attachment uses the third and fourth function then more conditions are needed to separate them. For the offline version bucket load estimation can be implemented. This can also be expanded to estimate the weight when pallets or timber claw is used.

A very interesting extension of this work would be to, instead of the existing pattern recognition algorithm, implement a neural network (section 2.1.4) or another machine learning structure, that is trained to identify pre defined patterns. To do this, more information about the wheel loaders behavior in different operations is probably needed.

The function that identifies pallet handling needs development so it can separate loading and unloading events. Investigation is also needed to determine why one of the pallet events blocks the algorithm from identifying bucket unloading. One solution may be to use hierarch order for the different identification functions to work through the data for event identification. The certainty for the identifications and the time for the identification for both timber and pallet needs some work.

Bibliography

[1] B. Frank, J. Pohl, and J.-O. Palmberg, "Estimation of the potential in predictive control in a hybrid wheel loader," *SICFP'09*, 2009.

[2] T. Nilsson, C. Sundström, E. Nyberg, Peter Frisk, and M. Krysander.

[3] R. Filla, *Quantifying Operability of Working Machines.* PhD thesis, Linköpings University, Sweden, 2011. ISBN 978-91-7393-087-1 ISBN 0345-7524.

[4] A. Pentland and A. Liu, "Modeling and prediction of human behaviour," *Neural Computation*, pp. 229–242, 1999.

[5] D. Mitrovic, "Reliable method for driving events recognition," *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEM, VOL.6 NO.2JUNE*, pp. 198–205, 2005.

[6] J. Enström and T. Victor, "Real-time recognition of large-scale driving patterns," *IEEE*, pp. 1018–1022, 2001.

[7] H. P. Chan-Chiao Lin and J. M. L. Soonil Jeon, "Control of a hybrid electric truck based on driving pattern recognition," *Proceedings of the 2002 Advanced Vehicle Control Conference, Hiroshima, Japan, September*, 2002.

[8] M. Ruotsalainen, J. Jylhä, J. Vihonen, and A. Visa, "A novel algorithm for identifying patterns from multisensor time series," *World Congress on Computer Science and Information Engineering*, pp. 100–105, 2009.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning.* 1 ed., 2006.

[10] J. Giarratano and R. Gary, *Expert Systems Principles and programming.* PWS Publishing Company, 3 ed., 1998. ISBN 0-534-95053-1.

[11] A. K. Jain, R. P. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on pattern analysis and machine intelligence, vol. 22, no. 1, January*, pp. 4–34, 2000.

[12] P. Antoniou, J. Holub, C. Iliopoulos, B. Melichar, and P. Peterlongo, "Finding common motifs with gaps using finite automata," *Implementation and Application of Automata*, pp. 69–77, 2006.

Appendix A

Calculated parameters

Table A.1: Table over calculated parameters and for which phase.

Parameter	**Phase**
Time [%]	In main cycle
Time [%]	In sub cycle
Time [s]	Load phase
Time [s]	Drive loaded phase
Time [s]	Unload phase
Time [s]	Drive unloaded phase
Gear (vector)	Load phase
Speed (vector)	Load phase
Gear (mean)	Load phase
Lift lever (vector)	Load phase
Liftspeed	Load phase
Tilt lever (vector)	Load phase
Tilt speed	Load phase
Accelerator pedal pressure (vector)	Load phase
Applied break (vector)	Load phase
Fuel rate (mean)	Load phase [1]
Time [s]	Drive (loaded) phase
Speed (vector) [km/h]	Drive (loaded) phase
Speed (mean) [km/h]	Drive (loaded) phase
Speed (max) [km/h]	Drive (loaded) phase
Speed (min) [km/h]	Drive (loaded) phase
Speed (median) [km/h]	Drive (loaded) phase
Gear (mean)	Drive (loaded) phase
Distance [m]	Drive (loaded) phase
Accelerator pedal pressure (vector)	Drive (loaded) phase
Applied break (vector)	Drive (loaded) phase

Fuel rate (mean)	Drive (loaded) phase [1]
Time [s]	Unload
Speed (vector) [km/h]	Unload phase
Gear(mean)	Unload phase
Lift lever (vector)	Unload phase
Liftspeed	Unload phase
Tilt lever (vector)	Unload phase
Tilt speed	Unload phase
Accelerator pedal pressure	Unload phase
Applied break (vector)	Unload phase
Fuel rate (mean)	Unload phase[1]
Time [s]	Drive (unloaded) phase
Speed (vector)	Drive (unloaded) phase
Speed (mean) [km/h]	Drive (unloaded) phase
Speed (max) [km/h]	Drive (unloaded) phase
Speed (min) [km/h]	Drive (unloaded) phase
Speed (median) [km/h]	Drive (unloaded) phase
Gear (mean)	Drive (unloaded) phase
Distance [m]	Drive (unloaded) phase
Accelerator pedal pressure (vector)	Drive (unloaded) phase
Applied break (vector)	Drive (unloaded) phase
Fuel rate	Drive (unloaded) phase[1]
Time [min]	Cycle
Fuel rate	Cycle [1]
Fuel rate (vector)	Cycle [1]
Ls pressure (min)	Cycle
Ls pressure (max)	Cycle
Ls pressure (mean)	Cycle
Total time [min]	All loading phases
Total time [min]	All driving (loaded) phases
Total time [min]	All unloading phases
Total time [min]	All driving (unloaded) phases
Time (mean) [min]	Sub cycle
Time (total) [min]	Sub cycle
Time [s]	Standing still
Time (total) [min]	Standing still
Distance (mean) [m]	All driving phases
Type of cycle	Cycles
Attachment	Cycles
Total fuel [l]	Cycles
Fuel rate (mean) [l/s]	Cycles

[1]If the signal exists.

Appendix B

Table of plots

Table B.1: Table over plots that the post algorithm can show.

Signals	**Specification**	**Type of plot**
Lift and tilt angular (normalized)	Load and unload events	Graph
Speed	Speed events	Graph
Lift and tilt angular (normalized)	Main and sub cycles	Graph
Time	Standing still, Main-, sub cycle and unknown	Pie chart
Time	LAC, SLC, Cleaning/Retry and standing still	Pie chart
Gear	Time per gear	Pie chart
Fuel rate	In cycles	Histogram
Lift lever	Distribution on lift lever in load/unload phases	Histogram
Tilt lever	Distribution on tilt lever in load/unload phases	Histogram
Accelerator pedal	Applied accelerator in main cycles	Histogram
Brake pedal	Applied brake in main cycles	Histogram
Vehicle speed	Velocity in main cycles	Histogram
Ls pressure	Average ls pressure in main cycles	Histogram

Appendix C

User guide

C.1 Directory structure

You need to have three directories, one that contains the algorithm, one that contains datasets and one where the results are saved. The three directories need to be on the same level. Figure C.1 illustrates a possible view over the directories structure.

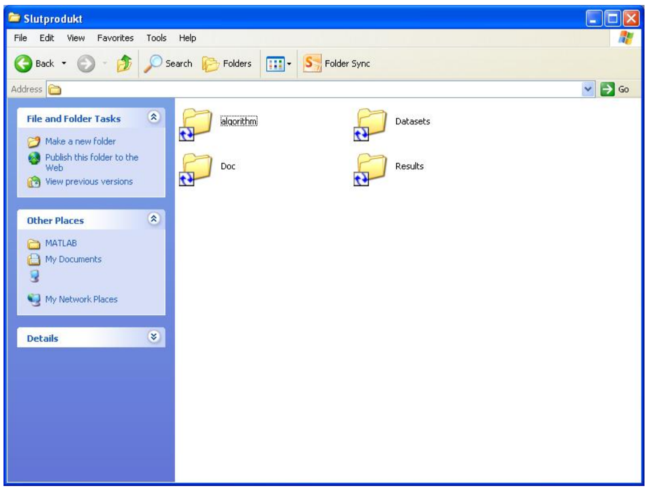


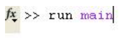
Figure C.1: Directory structure for the algorithm.

C.2 Add dataset

When data from a Wheel Loader are collected and converted to a .mat format, it should be added to the Dataset directory.

C.3 Start the algorithm

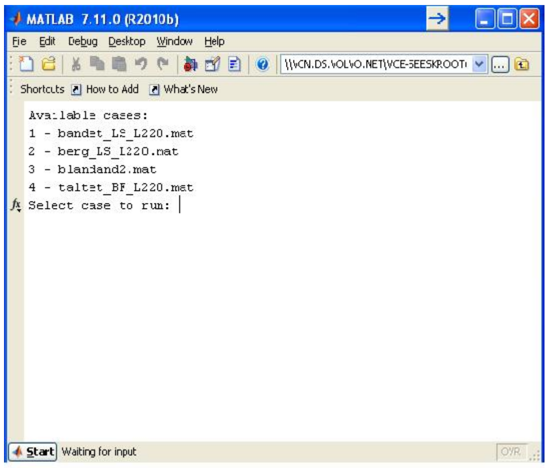
Open matlab, manoeuvre to the map called algorithm. Then write run main as in figure C.2.



```
>> run main
```

Figure C.2: Start command for the algorithm.

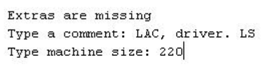
At first, it shows all available datasets that exist in the dataset directory in a list. Every dataset has a number and you choose which one you want to use by typing the number in front of that dataset. Figure C.3 shows what this process looks like.


MATLAB 7.11.0 (R2010b)
File Edit Debug Desktop Window Help
\\VCN.DS.VOLVO.NET\VCE-SEESKROOT
Shortcuts How to Add What's New
Start Waiting for input
OVR

```
Available cases:
1 - bandet_LS_L220.mat
2 - berg_LS_L220.mat
3 - blandand2.mat
4 - taltet_BF_L220.mat
Select case to run:
```

Figure C.3: Available datasets.

When a dataset is chosen, the algorithm asks you to type in a comment and the size of the machine. Figure C.4 shows how it looks.



```
Extras are missing
Type a comment: LAC, driver. LS
Type machine size: 220
```

Figure C.4: A comment and a machine size example.

Supported machine sizes are displayed in table C.1.

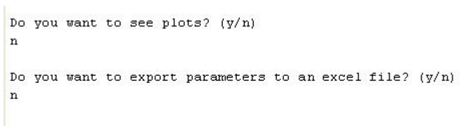
Table C.1: The machine (Volvo in production wheel loaders) models that the algorithm supports. Note that electric servo is needed on the machine.

90 [1]
120
150
180
220
350

[1]Needs extra sensors

C.4 Options

When the algorithm is completed with the pattern recognition and estimation of characteristic parameters, it asks the user the following questions; ”Do you want to see plots?” and ”Do you want to export parameters to an excel file?”. This is shown in figure C.5.



```
Do you want to see plots? (y/n)
n

Do you want to export parameters to an excel file? (y/n)
n
```

Figure C.5: Possible options to present the characteristic parameters.

The questions are answered with yes or no. If yes is chosen on the first question a number of plots are produced and displayed. If yes is the answer to the second question an excel document is created that contains characteristic parameters for the hole dataset in the first sheet. On the second sheet is every identified cycle presented and characteristic parameters for every phase.