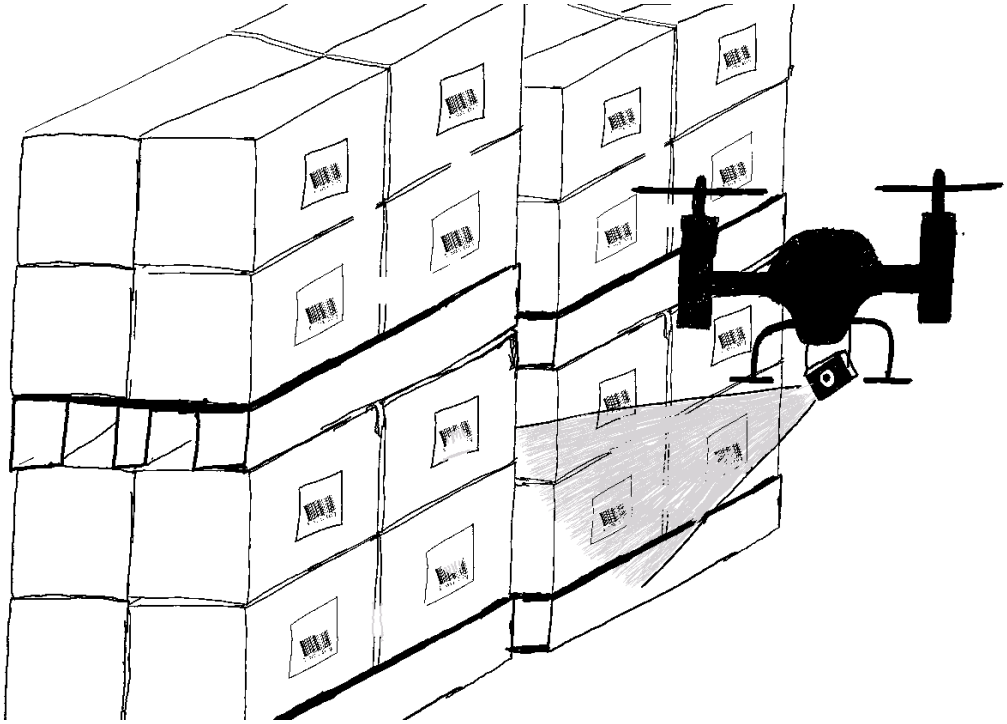
Cycle counting with UAVs

Sample selection in time-restricted scenarios with neural network predictions

Thijs G. Hammink
University of Twente



University supervisors
Martijn R.K. Mes
Dennis R.J. Prak

Company supervisor
Niek Tijink

External advisor
Rob H. Bemthuis

April 9, 2022

Management summary

The Autonomous Indoor Drone Applications research group (AIDA), is a collaboration between researchers from the University of Twente and the Aerobotic Tech Team Twente (A3T). This research group investigates various topics concerning indoor applications for UAVs. One of those topics is finding the means to deploy UAVs for fast and frequent cycle counts. During cycle counts, the goal is to check whether the physical inventory meets what is stated in the inventory records. When the physical inventory does not meet the stated inventory of a location, then a location's inventory record is considered inaccurate. Two main reasons why inventory counting is important are:

1. It gives the warehouse operator the possibility to update the inventory record such that it meets the physical inventory when inaccuracies are encountered during the count.
2. Inventory counting information gives the warehouse operator an estimate of the overall inventory record accuracy (OIRA).

Warehouses that are operated for 24 hours, 7 days a week, have limited time to execute cycle counts. This time is limited since it is required that the warehouse is not operated during the cycle counts in terms of safety. Therefore, cycle counts can only be operated during, e.g., short breaks of the personnel, which is around 15 to 20 minutes per break. This limited amount of available cycle counting time increases the importance of the decision which locations to count per cycle counting moment in comparison to overnight cycle counts, where it is possible to count the whole or large parts of the warehouse. This brings us to the following main research question:

"*How to design a generic (operational) cycle counting method for a single UAV with time restricted counting intervals?*"

We conducted a literature review to find insights into often used existing cycle counting policies. Here we found six main types of cycle counting policies. Each of these types focus on a specific part of the cycle count and have their own prioritisation rules for selecting certain items over others during a cycle count. For example, the ABC cycle counting policy prioritises items based on the cost of an inaccuracy and opportunity based cycle counting prioritises based on the ease of counting. From this literature review, we also found that certain cycle counting policies, assume a certain correlation for the occurrence of a mismatch between an inventory record and its physical inventory, based on certain warehouse characteristics (e.g., the transaction-based cycle counting). These cycle counting policies have found to be effective if this correlation is present in the warehouse.

Based on the findings of the literature review, we propose a cycle counting method that tries to learn the correlation between inventory record parameters and the occurrence of inaccurate records. The method predicts the chance that the status of a location is *inaccurate* and tries to construct a route such that the sum of the predicted chances of visited locations is maximised. The model is a greedy heuristic where we select the location that gives us the most prediction value per unit time cost. This time cost is the sum of additional travel time to add the location to the existing route and the scan time per location. This additional travel time is calculated using a cheapest insertion algorithm and the predictions about the chance that the location inventory record status is *inaccurate*, are made by a neural network.

We made a simulated warehouse based on a case study of Bolk logistics, where each day transactions are simulated. These transactions change the parameter values of the inventory records of each location. Based on these values, each day inaccuracies are simulated. After each simulated day, one cycle count is executed. For each cycle count a route is constructed using our proposed method. The route always starts and ends at the docking station (DS) of the UAV.

We used the simulated warehouse to conduct five types of experiments. The first type of experiments investigates various rules for selecting the first location (seed location) during the construction of the cycle counting route. During the second type of experiments, three methods for predicting the chance that the status of a location is *inaccurate* are compared to each other. During the third type of experiments a learning period is introduced, where various exploration and exploitation techniques are compared to each other. This learning period starts at the introduction of an UAV to a new warehouse and is used to improve the quality of the neural network predictions. The fourth type of experiments investigates the effect of the length of this learning period. Finally, a sensitivity analysis is conducted for various parameter settings. For each setting we compared the performance of our proposed method to a location-based (also known as progress-based) benchmark policy and a random sampling benchmark policy. From these experiments we conclude the following:

1. The selected seed location highly influences the direction of the constructed cycle counting route and which location neighbourhoods are visited. Especially, for locations further from the DS an appropriate seed location rule is required such that the locations are visited.
2. Using a neural network for predicting the chance that a location's inventory record is inaccurate, tends to give better predictions on the long-term in comparison to linear and logistic regression.
3. For all (except UCB-1) exploration-exploitation methods the OIRA drop during the learning phase, but the prediction errors after the learning phase are lower in comparison to not using a learning phase. For the UCB-1 method the OIRA increases during the learning phase but the prediction errors after the learning phase are not lower in comparison to not using a learning phase.
4. Using a special learning phase can help to increase the prediction performance of the neural network on the long-term. However, the cost of a gain of OIRA performance is high and the gain decreases over time.
5. (a) The proposed cycle counting method gives a comparable result in terms of OIRA performance to a benchmark location-based cycle counting policy for each of the sensitivity analysis experiments, but it significantly outperforms a pure random sampling policy.
   (b) The proposed cycle counting method significantly outperforms the location-based benchmark policy in terms of estimating the OIRA of the warehouse. It even tends to slightly outperform the random sampling policy in terms of estimating the OIRA.

Based on these findings, we conclude that our cycle counting method is able to learn the failure behaviour of a warehouse and to construct cycle counting routes based on these predictions. The performance of the model after the learning phase is comparable to our location-based benchmark policy. For example, we found that for counting intervals between 10-20 minutes the method found a comparable amount of inaccurate inventory records and had comparable OIRA results.

Unfortunaly, our more complex proposed model is not able to outperform the simple benchmark policy in terms of our objective to find as much as possible inaccuracies. We expect that the most important reasons why this is the case are the lack of spread in our simulated failure rate and the combination between the assumption that an inaccuracy stays inaccurate until it is visited and the definition of our objective function that does not take the length of an inaccuracy into account. Also, we treated each inaccuracy as equal. In practice, the severity of various inaccuracies are different.

Based on the previous paragraph and some other research limitations and findings, we propose four main research topics for future research. These topics are the objective function (1), the failure function (2), construction of the routes (3) and score models for the learning phase (4).

1. Redefine the objective function such that the length of an inaccuracy is penalised. Also the allocation of a cost factor to the inaccuracies makes the model more applicable in practice, due to the severity of an inaccuracy.

2. Testing the proposed cycle counting method in practice can overcome the big research limitation of the failure rate function as described above. We expect that this will also make it harder for the Neural network to learn the failure behaviour of the warehouse and this can increase the necessity of a learning phase.

3. We proposed five seed customer rules such that locations further down the aisles are selected. The seed customer initialises the direction in which a cycle counting route is constructed. During the

construction of the cycle counting route we want to maximise our objective by adding locations with the highest score-cost ratio. If the seed customer directs to a neighbourhood with low predictions, then our model can construct a route where our objective is far from optimised. We propose to research the effect of constructing multiple routes per cycle counting moment, where the route with the highest objective value is executed.

4. During the learning phase, each cycle count moment we assign a score to each location using an exploration-exploitation technique. By doing this, we assume that a location's failure rate depends only on the warehouse location. However, the failure rate of the warehouse depends on multiple parameter values, where the contribution of a certain parameter value to the failure rate is simulated constant. The location's parameter values change due to transactions in the warehouse and therefore the failure rate of the location also changes. Therefore, we propose to construct an exploration-exploitation model that predicts scores based on parameter values instead of the warehouse location, such that it is possible to reduce the model's uncertainty about the effect of the parameter values instead of calculating an uncertainty score based on its location in the warehouse.

Contents

1 Introduction **1**
1.1 Stakeholders . . . 1
1.1.1 The AIDA research group . . . 1
1.1.2 Bolk logistics . . . 1
1.2 Context description . . . 1
1.3 Research motivation . . . 2
1.4 Problem description . . . 4
1.5 Research questions . . . 5

2 Literature study **7**
2.1 Fixed cycle counting policies . . . 7
2.1.1 Random sampling . . . 7
2.1.2 ABC cycle counting . . . 8
2.1.3 Process control cycle counting . . . 8
2.1.4 Opportunity based cycle counting . . . 9
2.1.5 Transaction based cycle counting . . . 9
2.1.6 Location based cycle counting . . . 9
2.1.7 Conclusion . . . 10
2.2 Learning policies . . . 11
2.2.1 Deterministic optimisation vs stochastic optimisation vs learning . . . 12
2.2.2 Online and offline learning . . . 13
2.2.3 Counting policy based on supervised learning . . . 13
2.2.4 Bandit problem . . . 14
2.3 Sequential decision problem . . . 15

3 Use case - Bolk logistics **18**
3.1 Bolk Logistics . . . 18
3.1.1 Inventory counting . . . 18
3.1.2 Material handling . . . 19
3.1.3 Record-keeping process . . . 20
3.2 Transaction dataset . . . 20
3.2.1 Data collection . . . 20
3.2.2 Data exploration . . . 21
3.2.3 Data preparation . . . 21

4 Solution approach **24**
4.1 Design challenges . . . 24
4.2 Assumptions . . . 25
4.3 Mathematical model . . . 26
4.4 Inventory cycle counting method . . . 27
4.4.1 Step 1: calculate initial cost . . . 27
4.4.2 Step 2: calculate score value . . . 27
4.4.3 Step 3: seed customer . . . 29
4.4.4 Step 4: calculate additional cost . . . 29
4.4.5 Step 5: plan location with highest score/cost ratio . . . 30
4.5 Model addition: learning period . . . 31
4.5.1 The importance of good predictions . . . 31

4.5.2 The cost of learning . . . 31
4.5.3 Learning period . . . 32
4.6 Performance indicators . . . 32
4.6.1 Key performance indicators . . . 33
4.6.2 Performance indicators . . . 34

5 Simulation model **35**
5.1 Simulated warehouse . . . 35
5.2 Simulating inaccuracies . . . 36
5.2.1 Categories of errors . . . 36
5.2.2 Parameter selection . . . 37
5.2.3 Inaccuracy function . . . 37
5.3 Warm-up period . . . 38
5.4 Distance model . . . 39

6 Experiments **45**
6.1 Experiment settings . . . 45
6.1.1 Neural network settings . . . 45
6.1.2 Failure rate settings . . . 46
6.1.3 Physical environments . . . 47
6.1.4 Time settings . . . 47
6.1.5 Base model settings . . . 48
6.2 Individual experiment settings . . . 48
6.2.1 Seed customer . . . 48
6.2.2 Benchmark learning models . . . 49
6.2.3 Introduction of phase I . . . 49
6.2.4 Length of phase I . . . 49
6.2.5 Sensitivity analysis . . . 50
6.3 Experimental results . . . 52
6.3.1 Seed customer . . . 52
6.3.2 Benchmark learning models . . . 54
6.3.3 Introduction of phase I . . . 55
6.3.4 Length of phase I . . . 57
6.3.5 Sensitivity analysis . . . 58

7 Conclusion, discussion, and recommendations **62**
7.1 Conclusion . . . 62
7.2 Discussion and further research recommendations . . . 63

Appendices **68**

Appendix A Initial balance correction examples **69**

Appendix B Distance model verification examples **72**
B.1 Example 1 . . . 72
B.2 Example 2 . . . 73
B.3 Example 3 . . . 74
B.4 Example 4 . . . 76

Appendix C Code for implementation of the neural network model using Keras library **78**

Appendix D Failure rate distribution and chance of status *accurate* after n days **79**
D.1 Chance of location status *accurate* after n days given certain failure rate λ . . . 79
D.2 Distribution of λ for Base model A failure rate setting 1 . . . 80
D.3 Distribution of λ for Base model B failure rate setting 2 . . . 80
D.4 Chance of location status *accurate* after n days given certain failure rate λ . . . 81

Chapter 1

Introduction

In this chapter, we introduce Bolk logistics and the AIDA research group. This research is conducted on behalf of the AIDA research group. Bolk logistics is the company that provided a real-life case on which we implemented and tested findings of this research. An introduction on these parties is given Section 1.1. Next, we will give a context description about inventory counting in Section 1.2. Further, we will give a motivation for the necessity of this research in Section 1.3. Thereafter, we describe the problem that we want to solve with this research in Section 1.4. In Section 1.5 we describe the main research questions and additional research questions. For each of these additional research questions, we need an answer to be able to answer the main research question.

1.1 Stakeholders

In this section, we first introduce AIDA research group, the research group on behalf of which this research is conducted. Second, we introduce Bolk logistics, the company which provided a case on which we implemented and validated our proposed solution for our research problem. AIDA research group and Bolk logistics are consecutive introduced in Section 1.1.1 and Section 1.1.2.

1.1.1 The AIDA research group

The AIDA research group stands for Autonomous Indoor Drone Applications (AIDA) research group to which we will further refer to as AIDA. This group is a collaboration between Aerobotic Tech Team Twente (A3T) and researchers from the University of Twente. As the name suggests, this research group investigates the use of UAVs for different indoor applications. One of these indoor applications that they are investigating is to apply UAVs as a tool for inventory counting.

1.1.2 Bolk logistics

The story of Bolk logistics transport company (also known as Bolk logistics) began in 1934 with the establishment of the Looms & Bolk company [8]. Meanwhile, Bolk logistics has become a company with diverse activities, clients, and collaborations. Bolk logistics provides added value in multiple aspects of transport. The core business ranged from exceptional transport to transshipment to trailer transport. Bolk transports by road and by water.

In this research, we will focus on the warehouse of Bolk logistics, which is located in the harbour of Hengelo. This specific warehouse functions as a transit warehouse for pallet goods of consumable salt. This consumable salt is manufactured at a nearby plant which is located within a range of 2 kilometers of the warehouse. Bolk logistics provides logistic services for this manufacturer. These services are transportation to the warehouse, unloading, storage, and loading of the pallet goods.

1.2 Context description

Inventory counting is an important part of inventory management. Inventory counting is often done in order to check whether the physical stock matches the bookkeeping stock [36]. When the physical

stock does not match the bookkeeping stock, we consider this as an error. There are various inventory counting strategies. Some organisations choose to count their whole stock at certain times of the year. Other organisations choose to go for rolling stock takes where only a part of the inventory is counted per stock take, but these stock takes are executed more often in comparison to a full stock take.

Errors that are encountered during an inventory count could have happened during the whole interval of two consecutive inventory counts. A relatively large interval (e.g., months or years) makes it more difficult to find the cause of the error in comparison with an interval of, e.g., days or a weeks. In other words, the longer the time between consecutive counts, the longer the error exposure time, and thus the higher the number of expected candidates for the cause of the error [11].

As we will elaborate on further in Section 1.3, in this research we will focus on time restrictions. During our research we decided to focus on rolling stock takes, because the time required to do a full stock take is most of the times larger than the available time. For example, in the case of Bolk logistics the available time to do a stock count is around 15 minutes, while a full stock count takes more than 3 days. An often used method for rolling stock takes is cycle counting. Cycle counting is a sampling method where a part of a population is selected and measured [11]. This is called a sample of the population. Two important inferences can be made from these samples. The first is that the inventory record accuracy of the sampled items approximates the accuracy of the whole population of inventory records. The second inference concerns the type of errors found in the inventory records from the sample and their causes. Trying to find the underlying causes of errors is also known as sleuthing [11]. The power of finding these underlying causes is that corrective measurements could be made to prevent these errors from happening.

Classic cycle counting is executed by humans [11]. This is done by manually writing down product codes, by scanning barcodes, or by scanning RFID tags. The latter two could already be seen as a step towards semi-automation of cycle counting. A downside of a manual count is that it is subject to mental fatigue [16]. This could lead to loss of attention or inaccuracy. This loss of attention in combination with the use of forklifts and ladders could cause dangerous situations.

In literature we encountered emerging research into other execution techniques such as robots [50] and Unmanned Aerial Vehicles (UAVs) [16] [51]. These techniques focus on automation of the cycle counting process and make use of the established barcode sytem and/or RFID tag system. There are multiple differences between a barcode system and a RFID tag system, with both its advantages and disadvantages. However, they share a common character, both are made to be easily read by a computer, but are hard to read by a human.

1.3 Research motivation

As described in Section 1.2, we referred to examples of research into other execution techniques for cycle counting. In our research we will focus on UAVs. Flytbase Inc. [16] and Verity AG [51] both show that it is possible to count inventory with a UAV. In other words, it is shown that it is technical feasible to apply UAVs for the automation of inventory counting.

AIDA is also investigating means to deploy UAVs for fast and frequent cycle counts. Together with Bolk logistics they want to deploy a pilot study at the facility of Bolk logistics in Hengelo. So, at first, AIDA wants to realise automated inventory counting with UAVs for one of the warehouses of Bolk logistics. In the further future, the goal is to enroll the automated inventory counting with UAVs to multiple warehouses of Bolk logistics and also other logistic organisations. AIDA group eventually wants to provide the automated UAV inventory counting as a service. Therefore, we should try to find a generic solution for the problem which we will explain in Section 1.4.

To explain the necessity of this research, we have to understand the operational steps of an inventory counting process. To get an understanding of this counting process, we split the process into five steps. However, we first want to state that this is not a claim that in practice every inventory count follows these steps in the presented order. In Figure 1.1, we give a visualisation of the five inventory counting steps. During the first step, we will collect the current inventory records. In other words, we have to collect a list of the locations where each item is stored and the corresponding quantities of these items.

During the second step, we will select which items we will count during a specific cycle count and when we will plan when we will execute this cycle count. The third step is to determine the route we have to take to count the selected items for the cycle count. During the fourth step we will execute the count. In other words, we will follow the predefined route to the selected items and we will check whether the inventory record is accurate or inaccurate per item. The final step is the processing of the collected information. In case of a found error, we could correct the inventory record and we could conduct some sleuthing if we want to find the cause of the error.

(a) Collection of inventory records (b) Selection of which items to count at which moment in time (c) Determine the flying route the UAV is going to take

(d) Execution of the scheduled inventory count (e) Processing the results

Figure 1.1: Five typical operational cycle counting steps

As we briefly described in Section 1.2, and we elaborate on further in our literature study in Chapter 2, there are multiple ways to do inventory counting. However, in all of these methods we have to make the decisions which items we will count and at which moment in time we will execute these counts (Figure 1.1b). These two general decisions influence the performance of our inventory count. When the total available counting time decreases, the importance of these two decisions increases, which is also concluded in the master thesis of Veneman [50].

When we want to implement cycle counting with UAVs, we assume that the UAVs could only fly when the warehouse is not operated. The available time to fly could therefore be limited since we encountered that the available time that a warehouse is not operated could be limited. For example, the warehouse of Bolk Logistics is operated 24 hours a day for 7 days a week. This warehouse is not operated during short breaks of the personnel, which is around 15 to 20 minutes per break. Therefore research is required on inventory counting methods that could perform under time restrictions. This need for research into these time restrictions is also supported by the conclusions of Veneman [50] and Wijffels et al. [52]. In their research, they mention that certain methods would work better under time restrictions. However, they did not vary with time restrictions in their experimentation.

To conclude this section, the technology is ready to apply UAVs for automated inventory counting [16] [51]. However, more research is required on the selection of which items to count at which moment in time (step 2 Figure 1.1b). This is especially important in time-restricted scenarios [50]. Also, AIDA wants to apply automated inventory counting with UAVs as a service. Therefore, a generic solution for our problem is required, which we will discuss in Section 1.4.

1.4 Problem description

As described in Section 1.3, it is technically feasible to apply UAVs for automated inventory counting. However, we did not encounter anything about optimization of the inventory counting performance, while using UAVs. We will explain what we mean with inventory counting performance in this section. But first we will look into the overall goal of inventory counting.

Inventory counting is used as a tool to check whether the physical inventory meets the inventory that is stated in the inventory records [11]. We say that a location's inventory record is accurate if it matches the physical inventory (or is within predefined acceptable tolerances), otherwise we state that the inventory record is inaccurate. A high overall inventory record accuracy is desired, since many decisions are made based on the inventory records. By making these decisions (e.g., production planning or determining lead times for customer orders) it is assumed that the inventory that is stated on the inventory records is accurate. We could calculate the overall inventory record accuracy (OIRA) for a given warehouse by using Equation 1.1.

$$\text{Overall inventory record accuracy (\%)} = \frac{\text{amount of accurate records}}{\text{total amount of records}} \times 100\% \tag{1.1}$$

So we would say that a high OIRA is desired. We can achieve or maintain a certain level of OIRA by eliminating inaccurate records. We can do this by locating the inaccurate record and by taking corrective measures to make the record accurate. For example, if we want to achieve and or maintain an OIRA of 99% then only a maximum of 1% of the total amount of records is allowed to be inaccurate. One of the challenges is that we do not exactly know the total amount of inaccurate records. Only when we stop operations and we would take a full stock count (where we assume no counting mistakes), then we would have this information. And even if we had this information, then still this information becomes inaccurate when we start operating this warehouse again.

However, when we locate an inaccurate inventory record during an inventory count and we take corrective measures to make this record accurate, then the amount of inaccurate records will be lower. To maintain or increase our level of OIRA we have to balance between the amount of inaccurate records found and the amount of records that become inaccurate in a certain amount of time. The relationship between the amount of inaccurate records found, the amount of new inaccurate records, and the OIRA could be presented as follows:

(1)	Inaccurate records found $<$ New inaccurate records	$OIRA \downarrow$
(2)	Inaccurate records found $>$ New inaccurate records	$OIRA \uparrow$
(3)	Inaccurate records found $=$ New inaccurate records	$OIRA \rightarrow$

Now we know these relationships between corrective measures and inaccurate records in terms of OIRA, we can discuss the inventory counting performance as stated in the beginning of this section. As stated above, we can only correct an inaccurate inventory record when we find one. Only when we correct an inaccurate record this will give us a positive effect on the OIRA following the logic of the relationship described in the above paragraph. Therefore, we can express the performance of an inventory count in terms of the ability to find inaccurate records. We can make this ability measurable by, for example, counting the total amount of found inaccurate inventory records given a certain amount of time or measuring the time spend to find a given amount of inaccurate records or we can divide the total amount of inaccurate records found by the total amount of records counted. Independently of which of these three indicators we choose, the performance depends mainly on the choice of which items we count during the inventory count.

In Section 1.3, we encountered in literature that determining which items to count at which moment in time is an important step of the cycle counting process. Other steps are also important and can be optimized as well. For example, with the third step, determine the flying route the UAV is going to take, we can probably save some time as well. However, before we can determine a route, we first have to define the locations we are going to visit. Next to that, we can not find an inaccurate record for a specific item, if we do not choose to count it, but we can still find it if the route is not optimized.

To conclude, we can optimize the performance of the cycle counting procedure by making the decision of which item to count at which moment in time. The better the performance of the cycle counting procedure, the less time is required to find a certain amount of inaccurate records. Hence, we will spend less time on counting to improve or maintain our OIRA. Next to that, if we can achieve a high performance it will make the inventory counting solution with UAVs of AIDA more competitive towards other organisations that provide the same kind of solution.

1.5 Research questions

Based on the information from Section 1.3 and Section 1.4, we define our main research question as follows:

"How to design a generic (operational) cycle counting method for a single UAV with time restricted counting intervals?"

Next to the main research question, we define multiple smaller research questions. These research questions are used as input to give an answer to the main research question. We want to answer our main research question in a structured way. We will do this by giving an overview of the smaller research questions structured per chapter. At the end of each chapter we will give an answer to the research question(s) that correspond to that specific chapter. The research questions are structured such that we have to answer them consecutively. This is neccessary because an answer to previous research questions will be required as input to be able to give an answer to the main research question itself.

In Chapter 2, we start by looking into the literature to find suitable techniques, which we can apply for the selection of which items to count at which moment in time for our cycle count. Therefore, we will look into existing classical cycle counting methods, but we will also look into methods where the UAV can learn or make decisions based on counting experience. At the end of the chapter we should have a list of possible cycle counting techniques and we should have an overview of decision making techniques. The research questions that we therefore need to answers in Chapter 2 are:

1. Which main techniques for cycle counting are available?
   (a) Which inputs are required by these cycle counting techniques?
   (b) What are the advantages and disadvantages of these cycle counting techniques?
   (c) Which of these cycle counting techniques are applicable to inventory counting with an UAV?
2. How to select samples based on historic experience?

To get an understanding of the origin of inaccuracies between inventory records and physical inventory, we have to understand the record-keeping process of a warehouse. As mentioned earlier, Bolk logistics provides support to our research. For example, we use transaction data provided by Bolk logistics for a case study. By investigating Bolk's record-keeping processes, we get insight in how they keep track of their inventory. In this way, we get insights in how a record-keeping process is organised and we get an understanding of how to interpret the transaction data. In this research we assume that the record-keeping process and transactions of Bolk logistics are representative for other warehouses as well. Therefore, in Chapter 3 we need to answer the following research question:

3. How is a typical record-keeping process organised?

In Chapter 3 we identify the current situation at Bolk logistics and how their record-keeping process looks like and in Chapter 2 we found cycle counting and decision making techniques. With this information, we can develop a cycle counting method for UAVs in Chapter 4. Therefore we should answer the following research questions:

4. How should the cycle counting sample selection look like in a continuous learning environment?
5. How to evaluate the performance of the proposed method?

In Chapter 3 we identified the current record keeping process of Bolk logistics and we proposed a generic cycle counting method in Chapter 4. To evaluate our proposed method we want to run some experiments. To do so, we should set up an experimentation environment. As described above Bolk logistics provided data for a pilot study. Therefore, we should give an answer to the following research question in Chapter 5:

6. How can we set up an experimentation environment to assess the proposed cycle counting method using Bolk logistics as a pilot study?

In Chapter 5 we define and set up experimentation. In Chapter 6, we run various experiments to evaluate the performance of our proposed cycle counting method and we present the results of these experiments. Therefore, we should answer the following research questions:

7. How well can the proposed cycle counting method perform in time restricted scenarios for different error behaviours?

8. How does the performance of the proposed cycle counting method compare to other cycle counting techniques?

After we evaluated the various cycle counting technique, we conclude our research in Chapter 7. Next to that, we will give in Chapter 7 an extensive discussion on our research and give some recommendations for further research. In Figure 1.2 we present an overview of the topics of the chapters.

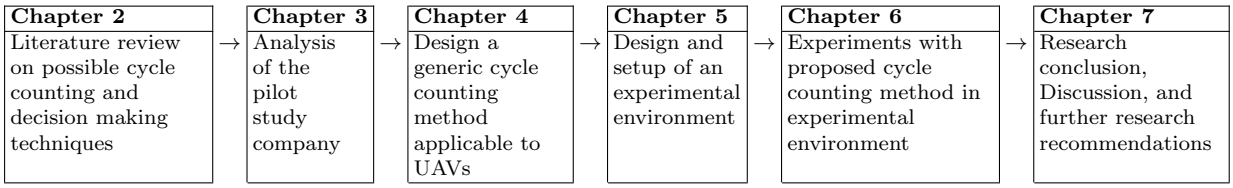


Figure 1.2: Overview of the research layout

Chapter 2

Literature study

In this Chapter, multiple cycle counting techniques are described. In Section 2.1 classical cycle counting techniques are described. Here the principles and the prerequisites of these cycle counting techniques are described. In Section 2.2, we describe what a learning problem is and what the difference is between *online learning* and *offline learning*. Next tot that, we present an offline supervised learning model for cycle counting based on historical data and an online learning model (k-armed Bandit problem). In Section 2.3 we dive deeper into policy learning for a specific type of learning problem, namely the sequential decision problem.

2.1 Fixed cycle counting policies

According to Brooks and Wilson, all cycle counting techniques are the same, with one exception: the method used to select a sample from its population [11]. To clarify this statement, it should be clear what is meant by a sample. A sample is a selection of certain members of a population. For example, when a sample with sample size 10 is taken from a population of 100 members (m), then ten members of the population are selected using the rules of the selected cycle counting technique. In Figure 2.1 an example of a sample is given. Here the sample selection rules of the random sampling technique are used, which are described in Section 2.1.1. In random sampling, the chance of an item being selected is equal for each member of the population. In other cycle counting methods, preference rules are implemented. These preference rules give some members of the population a higher chance of being selected in comparison to others. An example of such a preference rule is to give priority to items with a high total annual usage dollars in comparison to items with lower total annual usage dollars [43]. Such a preference rule makes that each method will select different members of its population (m) for the sample.

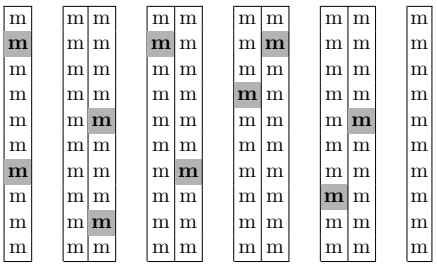


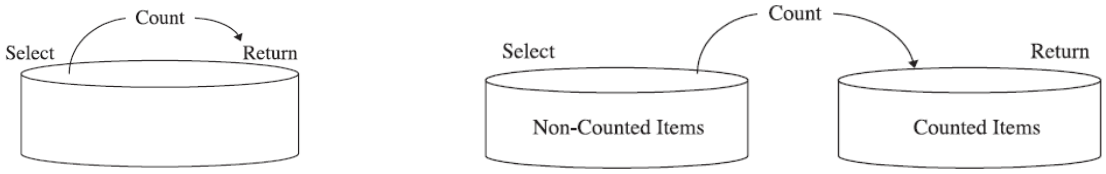
Figure 2.1: Random sample: sample size $n = 10$, population size $m = 100$, sample candidates $sc = 100$

According to Rossetti, Collins, and Kurgund [43], Brooks and Wilson [11], and Wijffels et al. [52] there are six main cycle counting policies. These six cycle counting policies are *Random sampling, ABC cycle counting, Process control cycle counting, Opportunity based cycle counting, transaction based cycle counting, and location based cycle counting.* These six cycle counting policies are described below.

2.1.1 Random sampling

In random sampling, a random sample from the population of inventory records is generated and the associated items are counted [43]. In random sampling the opportunity to be selected for the sample for

every member of the population is equal. There are two counting techniques that use random samples of parts: the *constant population* technique and the *diminishing population* technique [11]. The difference between the two could best be illustrated with an example. Consider the example of having a box with 1000 bolts. These 1000 bolts are the population. Each time we take a sample of 50 bolts to count. With the constant population technique the bolts are placed in the same box after counting. So, the next time we take a sample of 50 bolts there is a chance that the same bolts are selected to count. With the diminishing counting technique the 50 counted bolts are placed in a second box after they are counted. So the chance that the same bolts are selected for the next sample is zero. After all items from the first box are counted the same procedure is executed for the second box. The difference between these two techniques is also shown in Figure 2.2a and Figure 2.2b.



(a) Constant population [11, Figure 7.4a] (b) Diminishing population [11, Figure 7.4b]

Figure 2.2: population counting techniques

2.1.2 ABC cycle counting

Cost class	Part numbers (% of total)	Euros spent (% of total)	Times counted
A	20	80	4
B	30	15	2
C	50	5	1
TOTAL	100	100	

Table 2.1: example of an ABC classification based on percentage of total annual usage euros

ABC cycle counting is a special variant of random sample counting. The special thing behind ABC cycle counting is that it separates the population in different classes, where it is assumed that the importance of items is different among classes. In ABC cycle counting the population is stratified according to a Pareto analysis into three categories. The categories can be determined based on, for example, the total annual usage dollars, the frequency of issue, the length of the lead-time, or criticality of the product [43]. An example of an ABC classification based on the percentage of total annual usage euros is shown in Table 2.1. The idea of this separation is to assign different count frequencies to the classes, where items in class A should have a higher count frequency than items in class B and class C and items in class B should have a higher count frequency than class C.

2.1.3 Process control cycle counting

Process control cycle counting comes with two prerequisites [11]:

1. *Inventory records must have piece count by multiple location capability.*
   In other words, it should be possible to store information about the amount of items that are stored per location.
2. *An inventory record listing of all quantities in all locations for all parts is available to the cycle counter.*
   In other words, the counter knows the amount of items that should be in each location. So there is no blind count.

Process control cycle counting is a method that involves only counting items based on how easy the items could be checked [52]. This involves the *location* where the items are stored, *ease of counting* the items, and *obvious errors* [11].

During the cycle count it is checked whether every part number is placed in every location. However, when it comes to actual counting, only the easy parts are counted. Typically, these will be parts that are either low in quantity or packaged in ways that facilitate fast and easy counting [11].

Take, for example, a box with bolts. When the inventory record states that a thousand bolts are placed in this box, then it is checked whether this box is available at the stated location and if it looks like that there are thousand bolts. This is called an eyeball assessment. If it appears that there are thousand bolts, no physical count is made (a skip). When it appears that this is not the case (an obvious error), then the items need to be counted. For determining the inventory record accuracy, only items that are counted are considered in the calculation.

2.1.4 Opportunity based cycle counting

Items are count when there is a special opportunity. Many of these special opportunities are described in literature. A special opportunity is for example, counting during restocking of an item or when an item is reordered [52]. These counts are called opportunity counts because stock attendants capture them during their normal receiving and issuing of inventory, essentially capturing a free cycle count [11]. Another advantage of this is the often low amount of initial stock before restocking, which could save a lot of counting time.

2.1.5 Transaction based cycle counting

Transaction based cycle counting could be described as counting after a set number of transactions for a product [52]. Transaction based cycle counting comes as well with a prerequisite:

1. *A transaction history file of all transactions for all items is available.* In other words, the amount of times an item transaction took place should be known.

This method is especially effective for parts with a lot of transactions in a relative short amount of time. The amount of transactions on these parts makes it difficult to determine the cause of error when many transactions occur between counts. For example, if 100 transactions occurred between cycle counts, there would be 100 error candidates. This would make successful sleuthing difficult if not impossible [11].

Selecting a sample based on the amount of transactions of an item could help to limit the amount of error candidates. Take for example a population m of 100 items as shown in Figure 2.3. Only items that have more than five transactions could be selected for the sample. In Figure 2.3, a one represents an item with five or more transactions and a zero represents an item with less than five transactions. Counting the numbers one in Figure 2.3 results in 32 candidates for the sample (sc). When the sample size n equals 10, then 10 items are randomly selected from the list of 32 sample candidates sc.

1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0
1	0	**1**	0	1	0	1	0	1	1
0	1	0	1	0	1	0	0	0	0
1	0	0	0	1	0	0	**1**	1	1
0	0	1	**1**	0	1	0	0	0	0
0	**1**	0	0	0	0	**1**	0	**1**	1
0	0	0	0	1	0	0	1	0	0
1	0	**1**	1	0	0	0	0	1	**1**
0	1	0	0	0	0	0	1	0	0
0	0	0	0	**1**	0	0	0	0	1

Figure 2.3: Transaction based sample: sample size $n = 10$, population size $m = 100$, sample candidates $sc = 32$

2.1.6 Location based cycle counting

Location based cycle counting indicates a zone of products to be counted. Items that are placed together are counted at the same moment [52]. For example, counting all the items in a specific rack, aisle, or

zone. The main advantage of location based cycle counting it is counting items that are placed together. This method is especially efficient as it is counting as many items as possible in a given amount of time. In Figure 2.4, an example of selecting a sample from a population is given based on the location of the items. The selection rule in this example, is to only select items that are placed in zone 2.

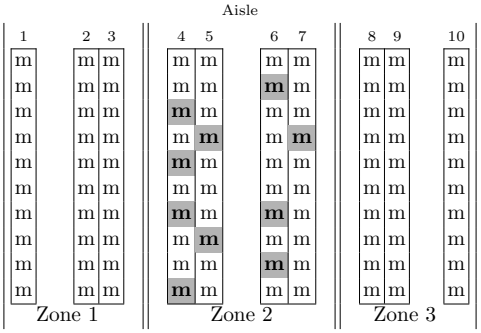


Figure 2.4: Location based sample: sample size $n = 10$, population size $m = 100$, sample candidates $sc = 40$

2.1.7 Conclusion

Based on this literature review, we can answer the first set of research questions:

1. Which main techniques for cycle counting are available?
   (a) Which inputs are required by these cycle counting techniques?
   (b) What are the advantages and disadvantages of these cycle counting techniques?
   (c) Which of these cycle counting techniques are applicable to inventory counting with an UAV?

In literature, we found six main cycle counting techniques. These techniques do not come with strict guidelines, but are described more as a general rules on how to select a sample from a population. For example, the ABC cycle counting technique, proposes to separate the population into multiple classes. There are multiple separation rules on how to determine these classes. Which rule performs the best depends on what the situation and what the goal is. This also holds for the other cycle counting techniques.

The difference between random sampling and the other five mentioned cycle counting techniques is that the five methods all make use of certain priority rules. For example, the transaction based cycle counting technique gives priority to items with a lot of transactions. This technique assumes that the higher the amount of transactions, the higher the chance that the records are inaccurate. This assumption could be true in practice, but does not particularly have to be the case. Lets assume a situation where this correlation is true, then this cycle counting technique is likely to be effective in finding inaccurate records and would outperform other cycle counting techniques such as random sampling, which do not account for this correlation. In a situation where this correlation is not true, it is possible that this transaction based cycle counting technique performs worse than random sampling. If errors are equally distributed over the population, then random sampling is likely to perform better.

The performance of a cycle counting technique could be described in terms of total found inaccuracies within a given amount of time or the ratio between total found inaccuracies and total items counted [52]. Beforehand, it is hard to determine which priority rules will perform best. An experienced warehouse manager who understands the causes of the inaccuracies of his or her warehouse, could probably make an educated guess about on which correlations a cycle counting technique should be based such that it would perform well. However, even with this experience it often takes a lot of trial and error [52]. In other words, there is not a generic cycle counting technique that is superior in performance compared to the other cycle counting techniques.

The selection of an inventory cycle counting technique that has good performance for a given warehouse is difficult. However, the results of a well implemented cycle counting technique are promising. For example, the book of Brooks and Wilson [11] shows multiple examples of companies that reached an

inventory record accuracy above 95 per cent after implementation of a cycle counting technique, while beforehand the accuracies were often below 80 per cent.

We think that opportunity based and process control cycle counting are not suitable for inventory counting with an UAV. These techniques make use of human assessments, such as the eyeball assessment or spotting an opportunity to combine certain tasks with inventory counting. The ABC cycle counting is less suitable for constructing a generic inventory counting model for an UAV, that focuses on eliminating inaccuracies in the warehouse inventory records. For making cycle counting routes the inventory record data needs to be separated into classes ABC and the model has to prioritize class A locations over class B and C, but the question is to what extent class A must be prioritised above class B and class B over class C.

Random sampling takes a random sample from the population. We think that this is not very efficient for eliminating inaccuracies in the warehouse, but we think it is the best technique for estimating the overall inventory accuracy of the warehouse, since it does not focus on a specific zone of the warehouse or on locations that are most likely to be inaccurate. Therefore, it is expected to give the best representative sample of the warehouse accuracy.

Location-based inventory counting focuses on the distance between locations, such that it can visit as much as possible locations during a cycle count. Equally to random sampling, location-based sampling does not focus on the likelihood that locations are inaccurate, but more locations are visited per cycle count. Therefore, we expect to find more inaccuracies with the location-based inventory counting method in comparison to random sampling, simply because more locations are visited. The transaction-based cycle counting method prioritises to visit locations which are more likely to have an inaccuracy in the inventory record. When the amount of visited locations is equal to location-based we expect that the performance of the transaction-based method is better in terms of finding as much as possible inaccurate records. However, we expect that the location-based method visit more locations in comparison to transaction-based. Therefore, we expect that there is a trade-off between the amount of locations visited and the likelihood that the inventory record status of a location is inaccurate.

2.2 Learning policies

We have to make the decision on which items we count at which moment in time. As described in Section 1.4, we want to count items that have an inaccurate inventory record. We want to count these items, because we can find inaccurate records if we decide to count them. We can also encounter the inaccurate records during operations, but this is something that we try to prevent by doing the inventory counts, such that it does not negatively effect operations.

So, we have to make the decision which items we count such that we find inaccurate records. Beforehand, we do not know if an inventory record is accurate or inaccurate. Hence, we need to make a decision under uncertainty [42]. However, as described in the sections 1.4, 2.1 and 2.1.7 we believe that there is a correlation between operations and error behaviour. In other words, we believe that the chance of having an inaccurate inventory record differs among items.

We want to make predictions about the chance of being inaccurate. In other words, we want to get a probabilistic understanding about the relationship between operations in the warehouse and error behaviour. Next to that, we try to collect new information about this relationship to improve our understanding. According to Powell and Ryzhov [42], a problem where you want to get and improve a probabilistic understanding about a relationship or function is called a *learning problem*.

In this section, we first explain in Section 2.2.1 what a *learning problem* is. Next to that, we use an example to show the difference between a learning problem, deterministic optimisation and stochastic optimisation. Thereafter, in Section 2.2.2, we discuss the difference between two learning approaches, *online learning* and *offline learning*. In addition, we provide an offline learning model for deriving cycle counting policies from data in Section 2.2.3. Eventually, in Section 2.2.4, we provide an online learning model. This online learning model is the (multi-)armed bandit problem, which is a frequently researched topic in optimal learning [42].

2.2.1 Deterministic optimisation vs stochastic optimisation vs learning

We now discuss an example to explain what we mean with a *learning problem* and to show what the difference is compared to deterministic optimization and stochastic optimization. Table 2.2 shows three sub-tables (Table 2.2a, Table 2.2b, and Table 2.2c). In each of these three tables there are four choices. Our objective is to choose the item that is most likely to be inaccurate.

Item	Value
1.	inaccurate
2.	accurate
3.	accurate
4.	accurate

(a) Deterministic optimization

Item	P(inaccurate)
1.	60%
2.	48%
3.	40%
4.	30%

(b) Stochastic optimization

Item	Initial P(inaccurate)	First Observation	Updated P(inaccurate)	Second Observation	Updated P(inaccurate)
1.	60%	accurate	40%		50%
2.	48%		48%	inaccurate	60%
3.	40%		40%		40%
4.	30%		30%		30%

(c) Learning

Table 2.2: (a) It is known which items are (in)accurate, (b) for each item it is known what the chance is of being inaccurate, and (c) for each item we update our belief (the chance that the item is inaccurate) of an item after an observation of that item

As is shown in Table 2.2a, we know that the first item is inaccurate, and the other three items are accurate. So, our best choice is to choose item 1. This choice is straightforward, since we know the actual values of the four items and only one item is inaccurate. In deterministic optimisation we assume that we know the actual values of our parameters without any uncertainty. In this example, it is easy to select which item we choose to count, but complexity can increase, when the total amount of inaccurate items increases. Each of these items come with corresponding coordinates and travel times between these coordinates. The complexity lies in selecting items and creating a route with selected items such that as much as possible inaccurate items are visited given a certain restricted amount of time.

In Table 2.2b, the same four items are shown. However, in stochastic optimisation we are not certain about our parameters, but we assume that we know the probability distribution of this uncertainty. For this example, the difference between deterministic optimisation as shown in Table 2.2a and stochastic optimisation as shown in Table 2.2b, is that for stochastic optimisation the labelled value (in)accurate is replaced by the chance that a certain item is inaccurate. Recall that our objective is to choose the item that is most likely to be inaccurate. As could be retrieved from Table 2.2b, the item that is most likely to be inaccurate is item 1. This does not mean that when we select item 1, that it will be inaccurate when we check the item. It can be the case that item 1 is accurate and the item with the lowest chance of being inaccurate (item 4) is inaccurate. Then afterwards we could say it was better to check item 4 instead of item 1. However, we do not know beforehand if it is accurate or inaccurate. Therefore, in stochastic optimisation we want to maximize our chance of selecting an item that is inaccurate and therefore we decided to select item 1.

In the first two examples, we knew the labelled value of the items or we knew the chance that a certain item is inaccurate or not. In a learning problem, we do not know the labelled values of the items and we also do not know the chance that a certain item is inaccurate. However, we can have a initial understanding of the chance that items are inaccurate. In Table 2.2c we took the values of the chances of the stochastic optimisation example as our initial values. In a learning problem we call these initial values our *belief state*. In Section 2.3, we explain what we mean by a *belief state*. For now we just assume that a belief state is a certain moment in time where we have a certain *understanding* of our parameters. In the case of our example, our initial *understanding* is the initial chance that a certain item is inaccurate. Based on our *belief state* we make our first decision. We would like to maximize our chance of selecting

an item that is inaccurate and thus we select item 1. We selected item 1, but we observed during our check that the item was accurate (Column 3 Table 2.2c). As stated at the beginning of this paragraph, in a learning problem we do not know the exact chance that a certain item is inaccurate. We observed that item 1 was accurate. In other words, we *learned* from our observation that item 1 was accurate. In a learning problem we use this information to update our *belief* about item 1, but also about features that are related to the observation of item 1. In other words, we can update what we observe, which can be more than only the inventory record status. In our example, we only observe the record status for item 1, so we only have to update the chance of being inaccurate for item 1. After updating (column 4 Table 2.2c), the chance of being inaccurate changed from 60% to 40% for item 1. In our new *belief state* (column 4 Table 2.2c), item 2 has the highest chance of being inaccurate, thus for our next observation we decide to select item 2. During our observation we found out that the item was inaccurate. So, again we update our *understanding* of the probabilities of being inaccurate. This results in the *belief state* shown in column 6 Table 2.2c.

So, the main difference between deterministic optimisation, stochastic optimisation and a learning problem is that in deterministic optimisation we know the values of our parameters. In stochastic optimisation we do not know the values of our parameters, but we know the probability distributions of these parameters and we do not update them after observations. In a learning problem, the values of our parameters are uncertain and the our understanding of the probability distributions is also uncertain. Therefore, in a learning problem we try to *learn* from *observations* to get a better *understanding* of the probability distributions of our parameters.

2.2.2 Online and offline learning

In literature we encountered that learning problems are often approached as either an online learning problem or an offline learning problem [41] [42] [52]. However, how the two concepts are interpreted in literature is sometimes conflicting. In this section we discuss the difference between online and offline learning and how we interpret these two concepts.

We could compare an offline learning setting to a setting where we have a budget to run N experiments, either in a simulation or a real-life environment. After we conducted these N experiments, we could use what we learned from these experiments to make the (according to our information) best decision possible [42]. In offline learning we are only interested on the performance of our final decision. Typical offline learning problems are for example, the testing of alternatives in a lab or in a simulation model. For example, when we have to visit 50 items for an inventory count in a warehouse and we want to know the shortest path to visit all of these 50 items, then we evaluate a lot of possible routes. However, eventually we are only interested in the route with the shortest path and we only follow this route for our inventory count. In other words, we first want to run some experiments and learn from these experiments to decide what our best solution is before we implement it into practice. Powell and Ryzhov [42] calls this distinct separation between experiments and implementation *the information collection phase* and *the implementation phase*. The first phase is *the information collection phase*, this is the phase where we are running experiments to learn, without regard to how well we are doing. After we finished *the information collection phase*, we continue with *the implementation phase*. During *the implementation phase* we use what we have learned to make our final design or choice, for which real costs are incurred.

Online learning could be compared to making decisions while running in the field [42]. In other words, we combine *the information collection phase* and *the implementation phase* such that we learn and implement at the same time. Recall, the example where we had to visit 50 items. In online learning we make a route and execute the route to check whether it was a good route or not. When it was a bad route, we actually spend the amount of time executing this route. Therefore, in online learning in contrast to offline learning, we care about intermediate results and not only about the best result.

2.2.3 Counting policy based on supervised learning

A more modern technique compared to the classical cycle counting techniques is cycle counting based on supervised learning [50]. Supervised learning or supervised machine learning is a type of machine learning [18]. This type of machine learning uses a known dataset with historical data. This dataset should consist of input variables (A) and response variables (B). The response variable (B) is the output that

you want the model to predict. In case of cycle counting, this could be a prediction whether an inventory record is accurate or inaccurate. The input variables (A), are variables on which the prediction is based. These could for example be, shelf life or the amount of transactions. In supervised learning mostly the historical dataset is split into a training set and a test set. The training set is then used to train the machine learning algorithm and the test set is used to evaluate the performance of the trained model.

Gitau [18] seperates supervised learning algorithms in two categories of algorithms. These are *regression algorithms* and *classification algorithms*. The main difference between the outcome of the two types of algorithms could be explained with the example above. In the example above two values for the response variable (B) where given. These where *accurate inventory records* and *inaccurate inventory records*. A classification algorithm outputs a discrete label. In other words, the output of a classification algoritm is either the label *accurate inventory record* or the label *inaccurate inventory record*. A regression algorithm on the other hand outputs a numerical value. In case of the example, the prediction result could be seen as the likelihood that a certain inventory record is accurate or inaccurate.

We consider the counting policy based on supervised learning, as offline learning. We consider this as offline learning, because there is a clear separation between the *information collection phase* and the *information implementation phase*. We will first start by collecting labelled data from practice. However, we derive this data from practice, we do not act while collecting this data. After, we collected enough data, we run our models on our collected data. We first start by training our models to get a good fit with the training data and then we test it on our testdata. Eventually, we only want to know what the best model is and from that model we will derive policies, which we could implement into practice.

Conclusion

Advantages:

- The model is updated when new counting data becomes available. Therefore, the model is able to adapt to changes in causes of inaccuracies.
- No expertise on which errors cause inventory record inaccuracies is required.
- Weights are assigned to features. These weights could help to identify error causes. When an error cause is known, it could be solved and eliminated, which could reduce the amount of errors over time.

Disadvantages:

- It requires historical data. To perform well it is data hungry.
- The amount of items that could be scanned in a given period of time is lower than location based cycle counting [50]. So the ratio inaccurate records found to the total amount counted has to be higher than location based.

2.2.4 Bandit problem

A k-armed bandit problem (k-AB) (also known as multi-armed bandit problem (MAB)) is a problem where we have k alternatives (arms) [29]. Each alternative has a different reward with an unknown probability distribution. The idea is that one of each of the alternatives has the best expected reward. Our goal is to find this alternative by repeated trials of the k arms. Each time we try an arm, we get to know something about the probability distribution of the reward of the arm we try. The more we try a specific arm the better our estimate of the real probability distribution of the reward. We call trying the arm for which we expect the best reward, given our current information, *exploitation*. However, we also have to play the other arms to get more information about the probability distributions of the other arms. We call this *exploration*. Each time we are going to try an arm we have to make the decision if we are going to explore or exploit. The balance between *exploration* and *exploitation* is the key trade-off for traditional k-armed bandit problems [5].

Within *offline learning* there is also a balance between *exploration* and *exploitation*, however in *online learning* this balance is more crucial. In *online learning*, every experiment that we conduct is also resulting in real costs. As described in Section 2.2.2, the intermediate results are more important in

online learning than in *offline learning*.

In a k-AB problem we want to know which of the k arms has the best expected reward. To learn which arm has the best expected reward, we have to play one of the k alternatives. Playing one of the arms at time t cost (C_t) us a certain amount of our total budget (B_t). Each time we play, we get a certain reward after playing (R_{t+1}). When we could add our direct reward to our budget, then the transition function of our budget from time t to time $t+1$ could be presented as given in Equation 2.1. So, the total times we could play depends on our initial budget, the cost of playing and our reward. A bad reward could limit our amount of learning possibilities. However, this does not mean that it was a bad decision. It could be useful to learn that a certain alternative k has a bad reward, because when we have no information about a certain alternative than we could not say anything about the expected reward of the alternative. However, when we already knew that it was a bad alternative, because we already measured the alternative frequently, then we could say it was not that useful to measure the alternative again. In a *learning problem* we call this usefulness the *value of information*.

$$B_{t+1} = B_t - C_t + R_{t+1} \tag{2.1}$$

Stationary rewards vs non-stationary rewards

In a classic k-AB problem the distributions of the rewards are considered stationary [5]. In other words, we assume that the distribution of the rewards does not change over time. If we encounter that the distribution of the rewards changes over time then we consider the distribution of the rewards non-stationary. The reason why we prefer stationary distributions is because we can make multiple observations and based on these observations we could say something about the expected reward for a next observation. When we have to deal with a reward distribution that changes (non-stationary) then our previous observations might not be reflecting the truth about the current state of the reward distribution [45].

2.3 Sequential decision problem

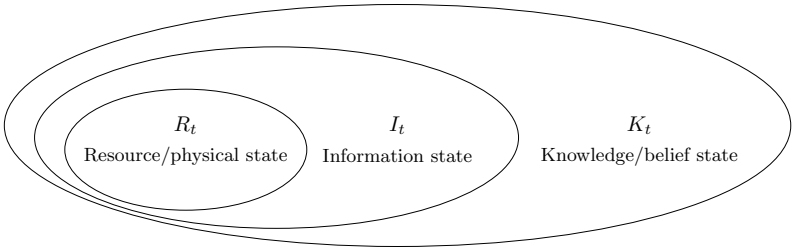
An example of a sequential decision problem is the Bandit problem given in Subsection 2.2.4 [24]. A sequential decision problem is a problem where we have to perform an *action* (e.g., a decision) in each *state* (e.g., moment in time), such that we find a good policy to meet our objective (e.g., minimisation of costs) [40]. In literature, we found a lot of different vocabulary and approaches for a sequential decision problem [40]. Powell [40] provides a framework to help us to make a decision on which type of approach we can take for our problem. In this section we will further introduce sequential decision problems and the corresponding framework of Powell [40]. Next to that, we will give a general formulation of our counting policy problem in terms of a sequential decision problem and we argue why our problem can be considered a sequential decision problem.

Elements of a sequential decision problem

A sequential decision problem consists of six elements [40]. We introduce these six elements one by one:

- *State*: this represents what we know at time t before we take an action. It is the minimally dimensioned function of history that is necessary and sufficient to compute decision function x_t.
  Mathematical representation: S_t
- *Actions*: this represents the decision[1] x we will make at time t.
  Mathematical representation: x_t
- *policy*: this represents the policy π on which we make the decision x_t.
  Mathematical representation: $X_t^\pi(S_t)$
- *Exogenous information*: this is variable information that we start to observe at time $t = 0$ (e.g., price, demand). We use the convention that any variable that is indexed by t is known at time t.
  Mathematical representation: W_t

[1] In other communities decision x is also referred to as a discrete action a or continuous controls u.



We could present the sequence that states (S_t), actions (x_t), and exogenous information (W_t) evolves over time t as follows:

$$(S_0, x_0, W_1, S_1, x_1, W_2, \ldots, S_t, x_t, W_{t+1}, \ldots, S_T)$$

- *Transition function*: the function that describes the evolution of the system from t to $t+1$
  Mathematical representation: $S_{t+1} = S^M(S_t, x_t, W_{t+1})$
- *Objective function*: We assume that we have a function that can be presented as a cost[2] function (C). This function may depend on the state S_t and the decision x_t. As an objective we want to find the policy π that minimizes the expected cost over a finite period of length T.
  Mathematical representation cost function: $C(S_t, x_t)$
  Mathematical representation objective function: $\min_{\pi \in \Pi} \mathbb{E}^{\pi} \sum_{t=0}^{T} C\left(S_t, X_t^{\pi}\left(S_t\right)\right)$

Classes of policies

A *policy* is a mapping on how we make a decision (action) in a certain state [40]. E.g., we have four alternatives from which we want to select the one with the lowest cost. Then we can define a cost function that describes the cost of each alternative. Since we want to select the alternative with the lowest cost, we select the alternative that has the lowest value resulting from the cost function. Here the cost function is not the policy, but the function makes it possible for us to select the cheapest alternative. According to Powell the combination of such a function and the decision is called a policy. Where Powell defines four classes of policies. These four classes are:

- *Policy function approximations (PFAs)*
- *Cost function approximations (CFAs)*
- *Value function approximations (VFAs)*
- *Look-ahead policies (LAPs)*

PFAs
A PFA represents some analytic function that does not involve solving an optimization problem. Examples of PFAs:

- (s,S) inventory model, where we learn the order up to level S and the reorder point s
- The direction at a particular intersection (routing logic), where we learn to go left, right or straight ahead

CFAs
CFAs are used in problems where a simple myopic policy can produce good results. A myopic policy is a policy that tries to maximises the immediate reward. CFA is an approximation of a cost function where your decision is made on the minimum outcome for the cost in a certain state. examples of CFAs:

- Shortest Path from a to b [41], calculate possible routes using a cost function and select the shortest path. We can learn this route, such that next time we have to go from a to b that we do not have to recalculate this route.

[2] A reward (R), or utility function is also used, but we assume that we can write these functions in terms of a cost function (C). E.g., max reward function: $\max \sum R\left(S_t, X_t^{\pi}\left(S_t\right)\right) = \min \sum -R\left(S_t, X_t^{\pi}\left(S_t\right)\right) \rightarrow -R(S_t, X_t^{\pi}(S_t)) = C(S_t, X_t^{\pi}(S_t))$

VFAs

VFAs are policies that try to make a prediction on the future based on the values of the post-decision state. An often used method for VFA is linear regression. The difference with PFA and CFA is that VFA looks at the decision that is made, PFA and CFA only look at the state that we are in. Next to that VFA is often used in situations where we have a lot of dimensions, the relationships are not as clear or easy as in a simple shortest path problem.

LAPs

The simplest look-ahead policy involves optimizing deterministically over a horizon H. In other words a look-ahead policy makes a forecast of the future/ what is coming next. Based on that forecast, decisions are made. For example, planning of capacity.

Hybrid policies

In addition to the four classes of policies, we can also combine these classes of policies. For example we could combine a LAP with a PFA. In case of a combination of these four classes, we call the policy a hybrid policy.

Chapter 3

Use case - Bolk logistics

This chapter discusses the company Bolk logistics, which collaborated with this research to perform a case study. In this chapter we first discuss some facts about the current inventory counts and the material handling at Bolk logistics in Section 3.1. Second, we introduce the dataset that Bolk provided for this research in Section 3.2. Next to that, we discuss how the dataset is modified such that it can be used to verify our proposed procedure of Chapter 4. We need to understand the material handling process of Bolk, such that we can understand the transaction dataset and retrieve the required data from the dataset. This is necessary since the dataset contains information of the whole warehouse (Bulk storage, (un)loading area, VNA buffer locations (IN and OUT), and VNA storage locations), but we want to clean the dataset such that we are eventually left with all transaction data concerning the VNA storage. The result of this chapter is a dataset that contains all VNA storage transaction data and is modified such that it can be used in Chapter 5 as simulated transaction data.

3.1 Bolk Logistics

As introduced in Section 1.1, we will focus in this research on the transit warehouse of Bolk logistics that is located in the harbour of Hengelo. In this warehouse, all kind of salt related products for the consumer market are stored. These products come from a near manufacturer and come to the warehouse as full pallet loads. So, in the warehouse only full pallet loads are stored. In the warehouse of Bolk logistics there are two types of storage. The first type is the Bulk storage and the second type is very narrow aisle (VNA) racking, where most of the items are stored in Bulk storage. In this section we first discuss some facts about one historic inventory count of the year 2020, such that we get some insight in their current inventory counting process. Thereafter we present the inventorying process of Bolk logistics. According to Brooks and Wilson [11] the inventorying process consist of two parts, the management of physical movements in the warehouse and the record-keeping of transactions. Below we present both parts seperately.

3.1.1 Inventory counting

Every year, Bolk conducts a stock count of the whole warehouse. This stock count is estimated to take around 64 hours for the whole warehouse. In the last stock count, 3.5 days were spend by two employees (56 hours) on manually counting items stored in the VNA. The VNA is 12.5 metres high. During the inventory count, one employee goes up in a cherry picker to write down the barcodes, while the other employee operates the cherry picker. The stock count for the Bulk storage is approximated to take 1 day for 1 employee (8 hours).

Based on an interview with two employees of Bolk Logistics, the capacity of the warehouse is determined to be around 15000 pallets. From this capacity it is estimated that 40% of this capacity is VNA racks and 60% Bulk storage. So, the capacity of the VNA is around 6000 pallets and the Bulk storage is 9000 pallets. The most optimistic mean counting time per pallet per employee for each of the two storage methods can be calculated by assuming that at the moment of counting the warehouse was full. We assume this, because we do not know the amount of inventory at the counting moment. The optimistic mean counting time can then be calculated by dividing the capacity by the amount of time spend per

employee. Note that in case of VNA, the total amount of counting hours per employee is equal to 28 hours. An overview of the current inventory counting statistics could be found in Table 3.1.

	Total counting hours	% of goods	% counting hours	capacity (No. pallets)	Mean counting time per pallet per employee
VNA	56	40%	87.50%	6000	16.8 sec
Bulk	8	60%	12.50%	9000	3.2 sec
Total	64	100%	100%	15000	8.64 sec

Table 3.1: Inventory counting statistics Bolk Logistics

As could be derived from Table 3.1, 87.5% of the total time spend on inventory counting is spend on counting only 40% of the pallet goods. There are three main reasons why the inventory counting time for the VNA racks is higher in comparison to the Bulk storage. The first reason is that two employees are required, where only one employee can count inventory. The second reason is that in Bulk storage in principle the same products are stored together per location, while there are much more VNA rack locations where in every location a different product can be stored. The third reason is the accessibility of the pallets. In the Bulk storage, pallets are easy to access and no tool such as a cherry picker is required. In the VNA racks, the pallets are much more difficult to access. The racks are 12.5 metres high and a cherry picker is required to see which product is stored at a specific location. In Figure 3.1a and 3.1b a picture of the Bulk storage and the VNA rack storage is shown.

(a) Bulk storage

(b) VNA rack storage

Figure 3.1: Storage methods used by Bolk logistics.

3.1.2 Material handling

The management of the physical movement of goods in a warehouse is also called material handling. According to Massey [31], material handling is a method that describes the movement within the scope of a building. In this part we present the material handling for the warehouse of Bolk logistics in Hengelo. To be more precise, we focus on the possible movement steps from arrival to departure of each pallet in the warehouse.

The process starts with an arrival of a pallet at an inbound dock. At the inbound dock the pallet is unloaded and placed at an unloading area. From the unloading area a forklift moves the pallet either to the Bulk storage or the VNA storage. The choice between Bulk or VNA storage is made by the warehouse management system. This system tells the forklift operator at which location a pallet must be stored. When the pallet needs to go to the Bulk storage it is placed directly at the appointed Bulk storage location. For the VNA storage the pallet is first placed at an ingoing buffer location (IN) at the beginning of the aisle in which the pallet needs to be stored. An extra step is necessary, because a special forklift which can operate in the VNA aisles is required. This special forklift picks the pallet from this buffer location (IN) and moves it to the appointed location.

A pallet stays at the appointed storage location until it needs to be shipped or placed at another location. When it needs to be shipped and it is stored at a VNA aisle, the pallet is moved from the VNA aisle to an outgoing buffer location (OUT) at the end of the aisle. From this buffer location (OUT) the pallet is moved to the loading area of an outbound dock. Finally, the pallet is loaded from the loading area into a truck at the outbound dock. For the Bulk storage there is no buffer outgoing zone. A pallet goes directly from the Bulk storage to the loading area of the outbound dock.

The decision to store a pallet either at the VNA or Bulk storage can be represented in the following rule of thumb. Large batches of the same product with a low throughput time are stored at the Bulk storage, and small batches with long throughput time, are stored at the VNA storage. In Bulk storage, pallets from the same batch are stacked together. When at a Bulk storage location the amount of pallets decreases it is possible that the remaining pallets are moved from the Bulk storage to the VNA storage. In this case each pallet is moved from the Bulk location to the buffer location for the appointed VNA aisle. A flowchart of the described possible paths for a pallet is given in Figure 3.2.

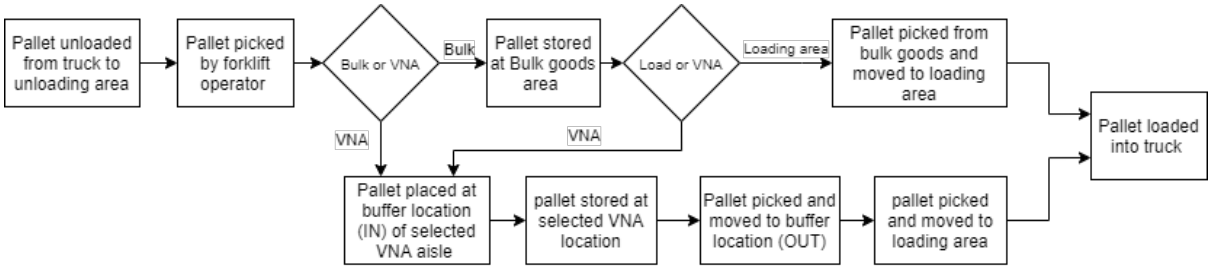


Figure 3.2: Material handling process

3.1.3 Record-keeping process

The material handling is how the physical pallets flow through the warehouse. In record-keeping, each physical movement within the warehouse is seen as a *transaction*. In other words, the transaction is a mirrored representation of the physical movement [11], where each transaction is represented as either an ingoing or outgoing transaction. Each location has ingoing and outgoing transactions, where an outgoing transaction from one location is the ingoing transaction for the other location.

The Warehouse management system of Bolk keeps track of the inventory at each location given in Figure 3.2. In other words, the system keeps track of the inventory at each individual Bulk storage location, VNA storage location, (un)loading area, and buffer location. To keep track of this, Bolk uses transaction data (Section 3.2) that is gathered during operations of the warehouse. Each individual movement is kept track of by scanning (or entering) both the location barcode and the pallet goods barcode.

3.2 Transaction dataset

In this section we describe the dataset that we use as input for our simulated warehouse of Chapter 5. This section is organised as following. First we discuss the data collection. Therafter, we present some statistics about the data set. Finally, we discuss how we adjusted the received dataset to the dataset as used in our simulated warehouse.

3.2.1 Data collection

From Bolk logistics, we received a dataset containing all transactions in the warehouse of the year 2019. This dataset consist of 978886 input rows of data. This dataset, is provided in excel format (.xlsx) and is retrieved from their warehouse management system.

3.2.2 Data exploration

The dataset consists of 978886 rows. Each row has 10 inputs. These inputs are Pallet ID, date, time of the day, warehouse hall number, location number, operator, last warehouse hall number, last location number, article number, batch number. In the dataset there are 341777 unique pallet ID numbers, In other words, 341777 different pallets are moved through the warehouse in the year 2019 with an average of 2,86 transactions per pallet.

The warehouse is operated 24 hours for 7 days a week. Only with Christmas the warehouse is not operated. In Figure 3.3, an overview of the weighted average amount of transactions (per week) for each day of the year is given. As can be seen in the graph, the weighted average of transactions per day is between 1900 and 3500. Except at the end of the graph, where the amount of transactions drops to around 1250. This can be explained by the fact that the warehouse is closed during Christmas which gives two days without any transactions. But those days count for the weighted average of the week.

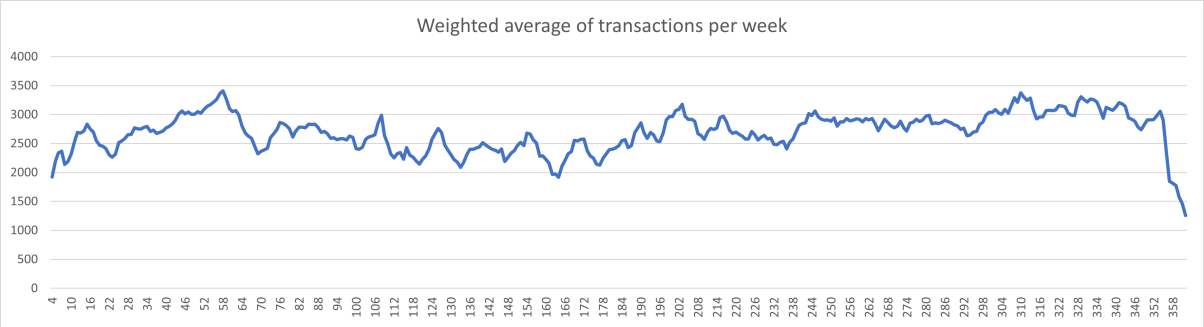


Figure 3.3: Amount of transactions per day of the year (weekly weighted average)

We can separate the locations into five categories. Namely, the Bulk storage, the VNA storage, unloading area, loading area, and other. In Figure 3.4, the distribution of the amount of transactions per each of the individual areas is given.

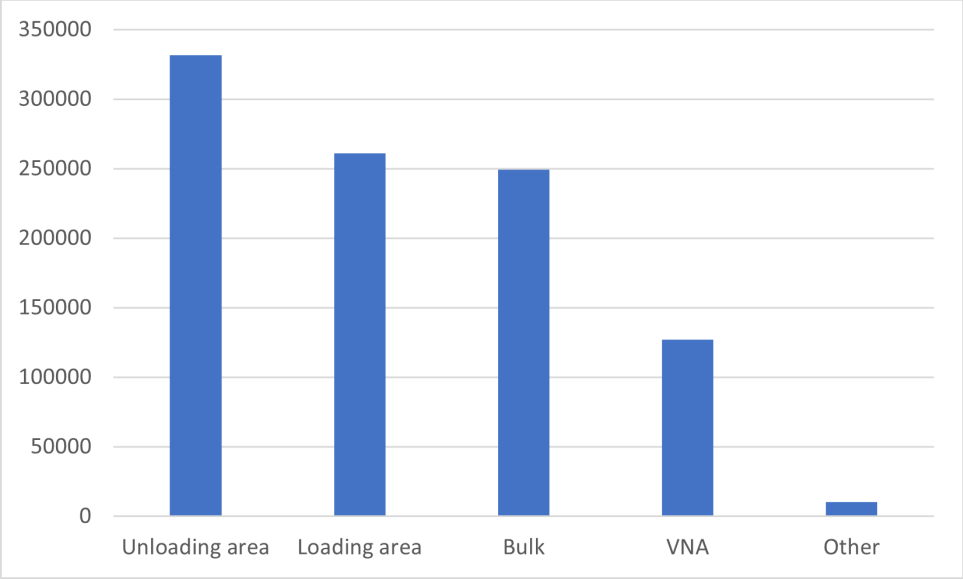


Figure 3.4: Distribution of amount of transactions per area

3.2.3 Data preparation

In our research we made a simulated warehouse based on the VNA rack storage of the warehouse of Bolk logistics in Chapter 5. This simulated warehouse uses the transaction data of each day of the above described data set to simulate transactions between consecutive cycle counting moments t. We refer to

the status whether or not a pallet is located at a location as the location *balance*. By backward reasoning we determined the *initial balances*. The initial balance is the balance of a location at the start of each simulation ($t = 0$). During each simulation we update the balances of each location between each cycle count moment t by using one day of transaction data. To be able to simulate transactions between cycle counting moments t we had to adjust the dataset with VNA transactions. Below, we first describe how we distinguished ingoing and outgoing transaction. Next we describe how we determined the initial balances, thereafter we describe some small required modifications to the transaction dataset.

Distinguish ingoing and outgoing transactions

In the dataset, each transaction has two location labels. Location label *Loknr* and location label $Vorig\_Loknr$. *Loknr* can be considered as the location it was going to and $Vorig\_Loknr$ as the location it was coming from. We filtered all the row of transactions where either *Loknr* or $Vorig\_Loknr$ had the value of a VNA storage location. We converted this filtered dataset to a dataset with only one location label, but with a special input label *type of transaction*. This value for this label is either *outgoing transaction* or *ingoing transaction*.

Initial balances

In our simulation we will compare the balances of the WMS of Bolk with our simulated warehouse. In our model we use all ingoing and outgoing transactions from and to a specific VNA location. From our dataset we extracted the initial balance of each location at the beginning of 2019 by using some rules of thumb. In this section we will briefly discuss how we determined the initial balances and which rules we used.

In our dataset for each transaction we have multiple inputs. Two of those inputs are current location and last location. If a VNA location is noted in the input field *currentlocation*, then we say it is an ingoing transaction for that location. If a VNA location is noted at the *lastlocation*, we say it is an outgoing transaction for that location. In this way we determined the amount of ingoing transactions and the amount of outgoing transactions per location.

Two other inputs, which are given in the transaction data, are the date and timestamp of the transaction. Using those inputs and the inputs described above we could determine per location whether the first transaction per location was an ingoing or outgoing transaction. We could do this, because we know that at least 0 pallets are present at the location and at most 1 pallet is present at the location. So, if the first transaction for each of the locations was an outgoing transaction we know our initial balance should have been 1. If the first transaction for each of the locations was an ingoing transaction, we know our initial balance should have been 0.

For more than 95% of all locations it was possible to use the above reasoning to determine the initial balance. For 265 locations it was not possible to determine the initial balance, because the amount of ingoing and outgoing transactions for those locations were zero. Bolk gave possible reasons for most of these locations. The reasons that were given were that certain locations must be left empty because it is not possible to place a pallet due to the limited height ($height = 8$) at certain locations or the location must be kept empty such that it could be used as safety passage ($Aisle = 20$). By analysing the 265 locations we found indeed that 123 of the 265 locations were located in aisle 20 and from the remaining 142 locations, 112 locations were locations at height 8. Based on the likelihood that these locations should be left empty we will set the initial balance of those locations to zero.

Next to the initial balance at the beginning of 2019 we also looked at the end and in between balances after each transaction of 2019. We did this by adding one to the initial balance for all ingoing transactions and subtracting one for all outgoing transactions from the initial balance. Then at each moment of the year the balance should either be zero or one. We found that for 5702 of the 5760 locations this was true. However, for 58 of those locations the end or intermediate balance was negative. Since it is not possible to have a negative balance we looked into all the individual transactions of these 58 locations. By analysing these transactions we found that for each of the locations there were multiple outgoing transactions in a row without an ingoing transaction in between and vice versa. This caused the negative balance or the too high balance, as could be seen for three of those locations in Appendix A.

we corrected the wrong balances, by either swapping time stamps of ingoing and outgoing transactions or by deleting some transactions such that the balance per location was always 0 or 1.

Other modifications

Now we discuss small modifications made to the data set. Most of these modifications were necessary to ease the programming of the simulation model. The made modifications to the transaction dataset are:

- After each count moment t, one day of transactions is simulated. Therefore, we extracted the day of the year number (range $1 - 365$) from the date label *datum*. This makes it easier to loop over the day numbers.
- The operator input is given as string value. The implemented learning model requires numerical inputs. Therefore, we changed each unique operator string value to a numerical input value.
- Each individual location is presented as unique location code. From this location code we extracted at which *place on shelf*, on which *shelf height* of which *Bay number* in which *lane of racks* a location is located. This is required for the distance calculation model presented in Section 5.4.
- The UAV travels through aisles, while pallets are stored in lane of racks. For the VNA storage of Bolk, each aisle consists of two lane of racks. Therefore, we assigned an unique aisle number to each pair of lane of racks.

More information about location codes and the assignment of aisles to specific lane of rack numbers is given in Section 5.4.

Chapter 4

Solution approach

In this chapter we present an inventory cycle counting model for UAVs, which is able to learn and make decisions based on observations. In Section 4.1, we start with a short recap of the found design challenges and our research goal. Next, we describe the assumptions we made in our modelling approach in Section 4.2. Thereafter, we provide the mathematical formulation of our problem as an integer linear program (ILP) in Section 4.3. This mathematical model has as objective to maximise the sum of the predicted chance of being inaccurate of locations visited each cycle count. In Section 4.4, we propose a step-by-step heuristic approach that constructs cycle counting routes, which focuses on maximising this objective. Thereafter, we propose an addition to the cycle counting model in Section 4.5. This addition focuses on the learning model which makes the predictions about the chance that a location is inaccurate. Next, we discuss the performance indicators, which can be used to measure the performance of the proposed model in Section 4.6.

4.1 Design challenges

In this research we want to design a generic cycle counting proceduce for a single UAV, which can perform under time restricted counting intervals. As described in Chapter 1, we assume that the failure behaviour of the warehouse is correlated to certain parameters of the operations in the warehouse. In Section 5.2, this failure behaviour will be discussed in detail. In Chapter 1 we also define the goal of cycle counting as searching for inaccuracies and the elimination of these inaccuracies to achieve and maintain a certain level of OIRA.

In Chapter 2 we discussed six main cycle counting techniques, which all but one came with their own priority rule. We concluded in Section 2.1 that this priority rule is effective when the by the priority rule assumed, correlation with the failure behaviour is true. Next to that, we found that a priority rule that is based on a correlation which is not true, is likely to perform worse in comparison to not using a priority rule (e.g. random sampling).

There is no general priority rule that works the best for every warehouse, as it depends on the failure behaviour of the warehouse. So, if we want to design a generic cycle counting method for a single UAV, then the method should be capable of learning the failure behaviour and establishing priority rules based on this learned failure behaviour.

Learning the failure behaviour does not automatically make the performance of the cycle counting method better. The method needs to make efficient decisions on which locations to visit during each cycle count. This can be done by an optimisation model, which optimises the visited inaccurate locations based on the likelihood of being inaccurate. This optimisation model has to use the learned information (likelihood of being inaccurate) as input. To be able to optimise as well as possible, we need to understand the failure behaviour as good as possible. This means that we have to learn the failure behaviour, to get the best performance on the long term, but on the short term we have to visit as much as possible inaccurate locations. In practice this means that we have to balance between exploring (learning) and exploiting (visiting inaccurate locations).

4.2 Assumptions

In this section we state the assumptions we made in our model. These assumptions are made to eliminate uncertainties which are not known and are not the main focus of this research. We made assumptions for three main subjects of our model, these are assumptions about time cost, assumptions about failure behaviour, and assumptions about the layout of the warehouse.

The first set of assumptions are about the amount of time certain tasks take for an UAV to execute these tasks and the amount of time an UAV get to execute these tasks. These assumptions are the following:

- The flight time is deterministic. The distance to be travelled is linear with the travelling time. We assume a constant flight speed without considering acceleration and deceleration.
- The scanning time per location is fixed.
- The available counting time per counting moment is fixed.

The second list of assumptions are about the failure behaviour of the warehouse. Inaccuracies can occur during operations, but they can also be encountered during operations. However, we assume that inaccuracies occur during operations, but they can only be resolved after being found in a cycle count. Therefore we assume the following:

- When an inventory record becomes inaccurate it stays inaccurate until it is visited during an inventory count. After being visited the inventory record becomes accurate.
- The failure rate for each location depends on the value of each individual parameter for that location. The parameters are information about transactions and information about the location. At each time moment t, the failure rate is calculated based on the values of these parameters (see Section 5.2 for more details).

The final set of assumptions are made about the layout of the warehouses. Our goal is to make a generic model that is applicable to multiple warehouses. However, we made some assumptions about the layout. We did this to keep the distance calculations of Section 5.4 as easy as possible. These assumptions are the following:

- An UAV could only enter and exit an aisle via the bottom entry/exit
- The warehouse consists of multiple aisles with identical width and height and an identical number of storage locations.
- Each storage location has the same width and height
- Inventory at each location is at most 1 item (Full) and at least 0 items (empty).

As mentioned above, the last set of assumptions are necessary, because of the distance calculation model we provide in Section 5.4. For example, when the width of each bay or the height of each shelf is not homogeneous, it is not possible to calculate the distance from each location to another location as given in Section 5.4. The choice to only allow the UAV to enter via the bottom entry of each aisle is made to reduce the complexity of the distance calculations (in comparison to e.g., flying over the shelves or multiple entry points) since distance calculation is not the main focus of this research. It is possible to extent our proposed cycle counting method by replacing the distance calculation model for a different more extended distance calculation model without having to adjust the rest of the proposed model.

Next to that, we made the assumption that inventory at each location is either one (full) or zero (empty). Having multiple items per location increases the modelling complexity. For example, multiple batch numbers, article numbers, and operators per location cause having multiple input values for each inventory record parameter. Our proposed learning method assumes that exactly one value is given for each selected learning parameter. Using multiple values per input parameter causes a calculation error. However it is possible to overcome such an error. This can be done by making the data suitable for our proposed learning method using data preparation techniques. For example, by replacing each combination of multiple inputs per parameter, to one unique numeric value. Then this unique value can be used as input for the learning method.

4.3 Mathematical model

In this section we present an integer linear program (ILP) of our cycle counting problem. We based the routing constraints of our ILP on a mixed-integer linear program (MILP) proposed by Anbuudayasankar, Ganesh, and Mohandas [3], which is a MILP for a vehicle routing problem with simultaneous delivery and pick-up with maximum route-length . Our objective is visit those locations, such that we maximise the sum of the predicted chance of being inaccurate per cycle count moment (Equation 4.1). The prediction values are considered input parameters for our ILP and are given by the learning model described in Section 4.4.2. Below we define the input parameters, the decision variable, the objective function and the constraints.

Parameter	Definition
R_i	Predicted chance of being inaccurate for location i
y_{ij}	Counting time cost to travel from location i to j
$STPL$	Scanning time per location
ACT	Available counting time

Decision variable	Definition
X_{ij}	$\begin{cases} 1 & \text{If UAV travels from i to j} \\ 0 & \text{Otherwise} \end{cases}$

Objective function

$$\max \sum_{i=0}^{N} \sum_{j=0}^{N} R_i \times X_{ij} \tag{4.1}$$

Constraints

$$\sum_{i=0}^{N} X_{ij} \leq 1 \qquad \forall j \tag{4.2}$$

$$\sum_{i=0}^{N} X_{ij} - \sum_{i=0}^{N} X_{ji} = 0 \qquad \forall j \tag{4.3}$$

$$\sum_{j=0}^{N} X_{0j} = 1 \tag{4.4}$$

$$\sum_{i=0}^{N} \sum_{j=0}^{N} (X_{ij} \times y_{ij} + X_{ij} \times STPL) \leq ACT \quad \forall j \tag{4.5}$$

$$X_{ij} \in \{0, 1\} \tag{4.6}$$

As described above, Equation 4.1 is the objective function which we want to maximise each cycle counting moment. Constraint 4.2 ensures that each location is visited at most once. If a location is visited by the UAV, then the location gets counted. Therefore we want the UAV to visit each location at most once per cycle count. Constraint 4.3 ensures that if the UAV arrives at location j, then it should also depart from location j. This restriction connects the locations and ensures that we get a consecutive route. Constraint 4.4 ensures that the UAV departs from the DS (i = 0). So, the combination of Constraint 4.3 and 4.4 ensures that the UAV always departs from and arrives at the DS. Constraint 4.5 ensures that the total travel- and scantime of the cycle count route does not exceed the available counting time. Constraint 4.6 ensures that the decision variable X_{ij} is either zero or one, since it is only possible to travel from i to j or not to travel from i to j.

4.4 Inventory cycle counting method

In a warehouse the total amount of locations is often large. For example, in the case of the VNA of Bolk logistics there are 5760 possible locations n, which can be visited in random order. When we want to compute all possible routes to connect these locations, then there are $N!$ possible routes [2]. We consider it computational unfeasible to compute all these possible routes to optimise our model such that we find an optimal route. The mathematical model of Section 4.3 has a time restriction which limits the length of the cycle count route. However, the amount of possible routes is still very large. Take for example the VNA of Bolk logistics where a cycle count route consist of 100 locations, then the amount of possible routes are $5760!/(5760-100)!$ [15]. The total amount of possibilities is too large to compute them all, such that we find an optimal value for our ILP. Therefore, we propose a heuristic approach, which limits the amount of possibilities.

We define each moment that the UAV counts the inventory as cycle counting moment t. Each cycle counting moment t a route needs to be constructed along selected locations. Each moment t the UAV starts at the DS. Starting from the DS locations are added one at a time until a full cycle counting route is constructed. This cycle counting route is then executed and information collected during this cycle count is then used as input information for the next cycle count. In Figure 4.1, the steps of our proposed heuristic approach are shown. Below we describe these steps for making a cycle counting route.

For every location $i \in I$ it is calculated what the minimum cost is to include the location to the cycle counting route (1). Next to that, a reward score of visiting the location is calculated (2). Then a first location is selected based on a predefined rule that focuses on a special character of the location (3). The first location that is selected based on a special character is called the seed customer [34]. After selecting the seed customer we calculate for each location $i \in I$, which is not the seed customer, the minimal cost to include the location into the route (4). This cost consist of travelling and scanning time. Thereafter, we add the location with the highest score-cost ratio given that the cost to include the location to the route is not more than the available counting time (5). We keep on adding locations to the route until there is no more available counting time (ACT) left. In the following sections we discuss each of the five steps in more detail.

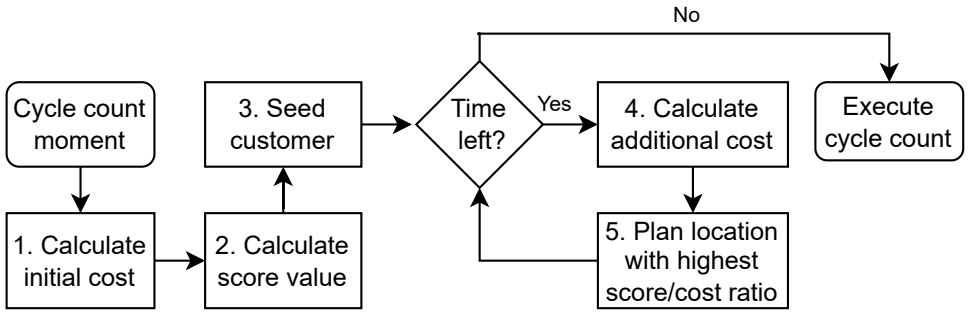


Figure 4.1: Inventory cycle counting method heuristic steps

4.4.1 Step 1: calculate initial cost

For each location $i \in I$ we calculate the travelling time from the DS to location i and back from location i to the DS. The exact distance calculations used in our simulation model of Chapter 5 are given in Section 5.4. The total initial cost per location is the sum of the travelling calculated travelling time and the scanning time per location (STPL).

4.4.2 Step 2: calculate score value

The mathematical model has as objective to maximise the sum of the predicted chance of being inaccurate of locations visited each cycle count. Therefore, we require a prediction model. In our problem we consider the inventory record status of each location as the dependent variable, which we want to predict. This status is either *accurate* or *inaccurate*. We want to predict the chance that the status is *accurate*

or *inaccurate*. Therefore, we approach the problem as a regression problem and not as a classification problem [17].

Since we have two status labels, we can choose to use a continuous or discrete regression approach. Both regression approaches assume the labels to be a numeric value. Therefore, we assign the value 0 to label *accurate* and 1 to label *inaccurate*. In a continuous approach, it is assumed that the status value can be every value in a specific range. In a discrete approach, it is assumed that there are a specific amount of status values. In case of using a continuous approach for our predictions, we can consider each continuous estimation value in the range 0 and 1 as the chance of being inaccurate for each location. In case of a discrete approach, there are two status values (0 and 1). For each of these two values a chance is calculated that the status is either 0 or 1. Then the chance of being inaccurate is the chance that the status variable has value 1.

There are multiple machine learning techniques which can be used for our regression problem such as linear regression, logistic regression, support vector machines (SVM) and neural networks (NN). We found that linear regression and SVM's are often used for predicting continuous variables [39] and logistic regression is often used for a discrete approach [9]. For NN's, we found that NN's are used for both kind of approaches, but that it depends on the configuration of the NN. For example, it is possible to configure a single node NN as a linear regression model [6], but also as a logistic regression model [1].

So, a NN can be used for both a discrete and continuous approach. Next to that, we found that NN's tend to outperform SVM's, logistic and linear regression at comparable prediction tasks [32] [28], given that there is as much training and computational power as possible. Given these above arguments, we decided to use a NN. However, in Chapter 6, we compare the performance of a few of the mentioned regression models by using our simulation model of Chapter 5.

In the remainder of this section we discuss details about the NN. First we briefly discuss the elements of a NN. Thereafter, we describe how the neural network is implemented and (re)trained.

A typical neural network consist of multiple layers. These are an *input layer*, *hidden layer(s)* and an *output layer* [46]. Each layer consist of multiple neurons. These layers of neurons are connected through *channels*. A channel can be considered as a weight, where the values of each neuron is multiplied with the weight of the channel. The sum of the multiplications of activated neurons are input values for the neurons of the next layer. This is often referred to as *forward propagation*. Whether or not a neuron is activated, and thus the value is propagated, depends on the *activation function*. An activation function can be seen as whether or not a certain threshold is met. There are multiple type of activation functions, but typically the type of activation function is equal per type of layer. Eventually all input data is propagated through the network. In our case outputs of the output layer will be predictions about the probability that the inventory record status is either 0 (accurate) or 1 (inaccurate).

We briefly discussed the NN above, now we explain how the NN is implemented in our proposed method. At the beginning of each cycle count moment we use the NN to predict the chance of being inaccurate for each location (1). We do this by inputting for each location the parameter information of the location. These predictions are considered the prediction score and are used in step 3 and step 5 of this heuristic to construct a cycle counting route (2). After each cycle count, we retrieve the status label (*accurate* or *inaccurate*) from the locations we visited (3). We combine this status label with the input parameters. After each cycle count all new count information is then added to the train data set. This train data set is used to retrain the neural network for the next cycle count (4). We do this, because we expect to get better predictions about the chance of being inaccurate when the amount of train data increases. The next cycle count moment we make predictions with the retrained NN. This implementation of the NN is also shown in Figure 4.2.

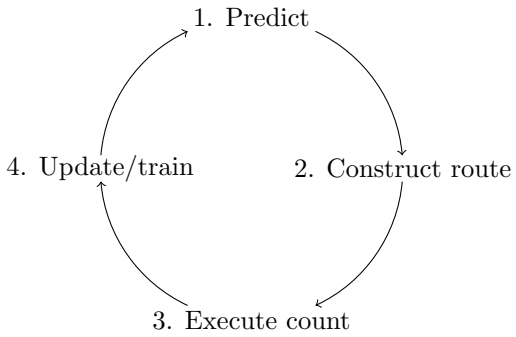


Figure 4.2: Implementation of the learning model in cycle counting process

4.4.3 Step 3: seed customer

We initialise each cycle counting route by selecting a seed customer. The seed customer is the first customer that is inserted to the route [23]. Commonly used seed customers in sequential vehicle routing problems (VRP) are farthest unrouted customer, customer with earliest deadline or earliest latest allowed arrival. For our problem we can not directly implement these seed customer rules, but based on these rules we propose the following rules for selecting a routes seed customer:

1. select location with the highest prediction score
2. select location where prediction score times the initial cost is the highest.
3. select top 10 locations with highest prediction score. Multiply these locations with the initial cost and select location with the highest value.
4. select top 50 locations with highest prediction score. Multiply these locations with the initial cost and select location with the highest value.
5. Randomly select a location from the locations list
6. Select location with the highest prediction score divided by the cost

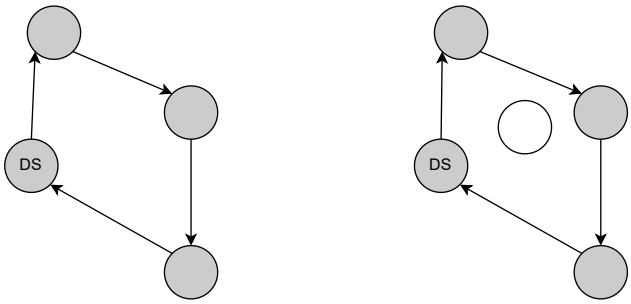
Rule number 1 makes sure that the location with the highest prediction score is visited regarding the cost to visit the location. This can be a location close to the DS or a location far away from the DS. Rule number 2 selects the location where the multiplication of score and cost is the highest. In other words, the rule prefers to select locations far away from the DS when prediction values are comparable. Rule number 3 and 4 are a combination of rule 1 and 2. The rule selects the set of locations with the highest prediction value, but prefers to select a location from this set which is further away from the DS. Rule number 5 does not prioritise based on initial cost or prediction values, it just selects a random location from the locations list. Rule number 6 focuses on the ratio score divided by cost and therefore tends to select locations close to the DS. In Chapter 6, we compare the performance of using each of these seed customer rules in multiple simulated environments.

4.4.4 Step 4: calculate additional cost

In step 1, we calculated the initial travel time cost for each location $i \in I$. In step 1 there is one routing option, to travel from the DS to location i and back. In step 4, we have an existing route such as for example shown in Figure 4.3a. When we want add an extra location to our existing sample of size n (e.g. Figure 4.3b), then we $(n+1)!$ options for constructing the route for the UAV. When the sample size increases the amount of options become too large to compute exact.

Therefore, we propose the use of heuristics for the construction of the route. We propose to use an insertion heuristic, which is a special category of construction heuristics [27]. The advantage of using an insertion heuristics in our problem in comparison to e.g., nearest neighbour is that it always calculates the cost of a fully connected route. This is advantageous, because we want to keep adding locations to the route as long as we meet Constraint 4.5 of our mathematical model.

An insertion heuristic starts with an existing tour consisting of multiple connected locations (e.g., Figure 4.3a). An insertion heuristic defines a rule between which two connected locations a new location should be added to the route. We propose to use the nearest insertion algorithm or for models with



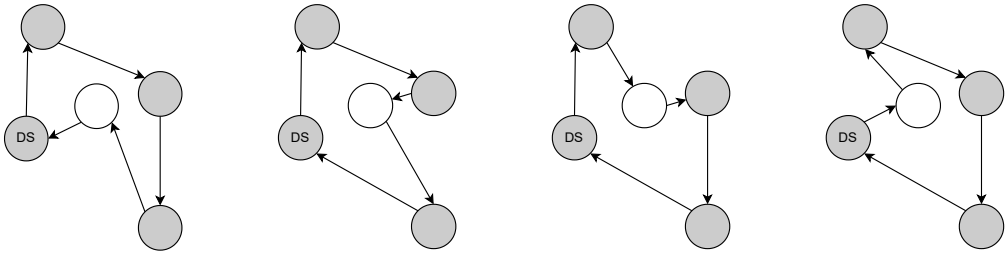
(a) Current route (b) Select new location

Figure 4.3: Nearest insertion example: existing route with 3 locations. Starting from and ending at the docking station

more complex cost models the cheapest insertion algorithm. We propose these heuristics since we want to minimise the additional cost of adding a location $i \in I$ which is not in the existing route. Due to our made assumptions about flight and scanning time (Section 4.2) the nearest insertion and cheapest insertion heuristic will give the same result for placing the location in the route, since the flight time is assumed linear with the distance. Therefore, we select the heuristic with the least computational complexity, which is the nearest insertion [27].

We illustrate the working of the nearest insertion heuristic by using the example shown in Figure 4.3. we select a new location (white node), which we want to add to our tour. In the figure, four grey nodes are shown which are connected by four arrows. The nearest insertion heuristic evaluates between which two connected nodes the new node must be placed, such that the additional routing distance (in comparison to the route with four nodes) is minimised. In Figure 4.4 we illustrated the four options for how the new tour with five nodes can look like.

As result from the nearest insertion heuristic we get the minimum additional travel distance for each location $i \in i$. Since we assume the travel time to be linear with the travel distance, we can calculate the additional travelling time. This gives us the total additional cost, which is the sum of the travelling time and the scanning time.



(a) Option 1 (b) Option 2 (c) Option 3 (d) Option 4

Figure 4.4: Nearest insertion example: 4 options for adding new location to the route

4.4.5 Step 5: plan location with highest score/cost ratio

Each cycle count moment we start at the DS and we determine a seed customer based on our seed customer rule of step 3. This results in a seed tour between the DS and the seed customer. In step 5 we add locations to this tour sequentially. Step 4 and step 5 are recurrent consecutive steps after adding a location to the tour. This is required since the additional cost to add a location to a tour changes. It depends on the closest two consecutive locations of the tour, which can change when locations are added to the tour.

The location which we add to our tour is the location which has the highest score-cost ratio and adding this location does not violate the ACT constraint 4.5. In this way, we want to maximise our objective

as defined in Section 4.3. This is a greedy approach where we select the location that gives us the most value (Ri) per unit cost (time) [19]. We keep on adding locations to our tour until it is not possible to add any more locations without violating the constraints.

4.5 Model addition: learning period

As described in Section 4.3, our objective is to find as much as possible inaccurate locations such that we get an as good as possible OIRA performance over time. We do this by trying to construct cycle counting routes that maximise the sum of the chance of being inaccurate. In this section, we first discuss the importance of the quality of these predictions and the cost of learning. Thereafter, we propose to combine the predictions of the learning model of step 2 of the proposed heuristic with score values given by algorithms that are designed for Bandit problems for a limited period, which we call the learning period. We propose this learning period to increase the quality of the predictions. We propose to implement this learning period at the introduction of an UAV to a new warehouse, since quality of predictions is expected to be the worst at the start.

4.5.1 The importance of good predictions

When an UAV is assigned to a new warehouse the UAV has no understanding of the failure behaviour of the warehouse it is assigned to. To get an understanding of the failure behaviour, the UAV has to execute cycle counts to collect counting information. The counting information of each cycle count can be used to learn about the failure behaviour. We expect that in the beginning where the amount of counting data is scarce that the predictions are not very accurate. These predictions are a certain belief of how failures are correlated to certain warehouse parameters. As mentioned in Section 4.1, a priority rule is effective when the by the priority rule assumed correlation with the failure behaviour is true. In other words, when our prediction model assumes a wrong failure behaviour, then our model proposed in Section 4.3 is not very effective. So, the effectiveness of our model depends on how well we can predict the chance of being inaccurate for each location.

4.5.2 The cost of learning

Learning does not mean that we always select locations from which we think that they are likely to be inaccurate. It can be vice versa that we decide to visit a location from which we think it is unlikely to be inaccurate but we want to get information from this location. So, we say that during learning our goal is not only to eliminate as much as possible inaccuracies, but to gather as much as possible understanding of the failure behaviour of the warehouse as well.

When we let an UAV focus on learning during a cycle count instead of a focus on eliminating inaccuracies, then there is a direct cost of warehouse accuracy. During learning the objective is not to maximise the sum of the PcI, but to reduce uncertainty in the predictions. This can cause a drop in OIRA. We think it is not desirable that the OIRA in a warehouse drops sharply or to drop below a certain threshold, since many warehouse operations use the inventory record information as input. Therefore, we have to keep the performance of our cycle count in mind, while we learn about the failure behaviour. In other words, learning comes with direct costs, but making these costs can be beneficial on the long-term. It is important to determine the length of a learning period where the benefits of learning outweigh the costs of performance.

A learning curve can be used to visualise this trade-off between the increase in prediction quality and the length of the learning period (Figure 4.5). A learning curve is a correlation between performance on a task and the number of attempts or time spend on the task [49]. The time spend on learning decreases the available counting time that can be spend on exploiting. Therefore, we want to switch from a focus on learning phase to a focus on exploiting, when the time cost to increase one unit in performance becomes too high. For example, when we want to get the best OIRA performance after a year of cycle counting. Then we have to balance the time we spend on learning and the time we spend on performance (eliminating as much as possible inaccuracies), such that we get the best OIRA at the end of the year.

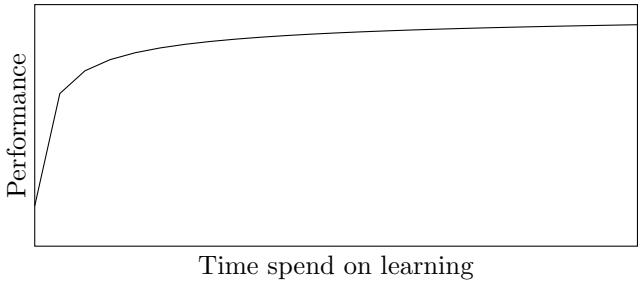


Figure 4.5: Learning curve

4.5.3 Learning period

Here, we discuss the learning period, which we further refer to as phase I. We call the period after phase I, phase II. During phase II cycle count routes are constructed as described in Section 4.4. The heuristic proposed in Section 4.4 consists of five steps. During phase I, we adjust the score value calculation of step 2, while the rest of the five steps heuristic approach is equal to Section 4.4.

The performance of the learning models proposed in Section 4.4.2 depend on the way these models are compiled (1), fitted (2), and the input data on which they are fitted (3). The third factor is directly influenced by historic cycle counts. The diversity and the amount of data that is inputted into the learning models influences the quality of the predictions. When the UAV is assigned to a new warehouse, then it starts without any counting data. At such a moment, the proposed learning methods are not likely to give good predictions, because it has (almost) no data to train on. In the basic heuristic (only phase II), we would decide which locations to visit by using these predictions as score value. In other words, we decide to use these low quality predictions for a pure exploitation strategy (decisions based on current best information) [7].

Above we discussed the effect of the quality of the predictions and the cost of learning. The discussed trade-off has similarities with the trade-off between exploration and exploitation in a Bandit problem, which we discussed in Section 2.2.4. In Bandit literature, multiple techniques are discussed that can be used to balance exploration and exploitation. The effectiveness of these techniques depend on the origin of the Bandit problem. The possible reward for visiting a location (accurate or inaccurate) in our problem can be compared to a special type of Bandit, namely the Bernoulli Bandit [30]. In a Bernoulli Bandit problem the player receives a binary reward (0 (no reward) or 1 (reward)) for playing an arm.

We propose to use techniques used for Bernoulli Bandits to calculate the score value per location during phase I. Three of those possible techniques are Thompson sampling, UCB-1, and ϵ-greedy [53]. In Chapter 6, we compare the introduction of a phase I, where either of these three techniques is implemented, to not using these techniques (directly start with phase II). Next to that, we use the technique that performs the most promising for determining an advise for the length of phase I based on the trade-off between the cost of learning and the reward of having good predictions.

During phase I, we implement and use these Bandit techniques the same way as we implement the learning model as shown in Figure 4.2 (Step 2). During phase I, we propose to let the learning model of phase II update, train, and predict such that it is possible to evaluate the development of predictions of the learning model to counting data gathered during phase I. We use the prediction score given by the selected Bandit technique to determine which data is gathered during each cycle count.

4.6 Performance indicators

In this chapter we developed a generic method for cycle counting with UAVs. In this chapter we focused on the balance between learning and the performance of each cycle count. Since this is a generic model, we want to provide multiple performance indicators such that it can be used to focus on different types of performances of our proposed method. We first discuss our key performance indicators (KPIs), which focuses on the overall effect of the cycle counts on the warehouse performance and on our objective of Section 4.3. Thereafter, we discuss some other important performance indicators, that focus on specific

aspects of the cycle count performance. Lastly, we provide some performance indicators that focus on the learning performance of our model.

4.6.1 Key performance indicators

Our main overall performance indicator for the warehouse is the OIRA, which we discussed in Chapter 1 and is shown in Equation 1.1. The OIRA represents the exact accuracy of the warehouse at a certain moment in time t. Therefore, we can also put in a time element t as given in Equation 4.7, such that we can keep track of the OIRA over a time period from $t = 0$ until $t = T$ where T equals the length of the total time span which we measure.

$$OIRA(t) = \frac{\text{Amount of accurate locations}(t)}{\text{Total amount of locations}} \times 100\% \tag{4.7}$$

In a simulated environment we can perfectly measure the OIRA. Since we need the status (*accurate* or *inaccurate*) of each location at each moment t. When we execute our proposed method in practice, then we only get status information about locations we visited at each moment t. Therefore, we propose an OIRA score based on historic measurements (MOIRA) and an OIRA score based on predictions of the learning model (POIRA). The MOIRA is the ratio accurate locations found and total locations count, over a certain time period Δt. So, if we want for example the MOIRA between $t = 5$ and $t = 10$ then we divide the amount of counted locations with status *accurate* by the total amount of locations counted as given in Equation 4.8.

$$MOIRA(\Delta t) = \frac{\text{Accurate locations counted}(\Delta t)}{\text{Total locations counted}(\Delta t)} \times 100\% \tag{4.8}$$

The POIRA is equal to 1 minus the predicted fraction of inaccurate locations. The numerator of the fraction is calculated as the sum of the predicted chances of locations having the inventory record status *inaccurate* (E[P(inaccuracy)]). These predictions are given by the learning model of Section 4.4.2. The denominator of the fraction is equal to the total amount of locations in the warehouse. The POIRA at moment t can be calculated as given in Equation 4.9.

$$POIRA(t) = 1 - \frac{\sum_{i=0}^{N} \text{E[P(inaccuracy)}_i](t)}{\text{Total amount of locations}} \times 100\% \tag{4.9}$$

We discussed three performance indicators for the accuracy performance of the warehouse. Each of the three indicators can be used in a simulated (offline) environment and the latter two can be used in an online environment, since the latter two do not require perfect information about the status of each location at each moment.

Above we presented KPIs for measuring the accuracy of the warehouse, which is influenced by the cycle counts. However, the presented KPIs do not tell much about the performance of the proposed cycle count method per individual cycle counts. The objective of our mathematical model in Section 4.3 is to maximise the sum of the predicted amount of inaccuracies visited per cycle count (Equation 4.1). We use this objective function as our main KPI for measuring the performance of each individual cycle count. Next to that, we want to measure the amount of actual inaccuracies found per cycle count, such that we can calculate an error rate between actual performance of the cycle count and predicted performance. With this error rate we can evaluate the ability of the learning model to make good predictions.

In other words, we propose one KPI for measuring the actual performance of each cycle count, one KPI for the expected performance of each cycle count, and one KPI for measuring how well our expectations meet reality. These three KPIs are consecutive amount of inaccuracies found per cycle count moment (FpM), expected amount of inaccuracies to be found per cycle count moment ($E[FpM]$), and the relative error between expected and actual amount of inaccuracies found ($RE[FpM]$)[20]. This $RE[FpM]$ can be calculated as given in Equation 4.10.

$$RE[FpM] = \frac{E[FpM] - FpM}{FpM} \tag{4.10}$$

4.6.2 Performance indicators

In the previous section we presented KPIs for measuring the performance of the cycle counts. These KPIs focus on the general performance of the cycle counting method. In this section, we present performance indicators that focus on the performance of a certain aspect of the cycle count method. We made a split between performance indicators for measuring the cycle counting performance and performance indicators for measuring the learning performance. We first discuss the cycle counting performance indicators (CPIs) and thereafter the learning performance indicators (LPIs).

Cycle count performance indicators

Whether or not finding 10 inaccuracies during a cycle count is a good performance depends on multiple factors. Such as the total amount of locations it can count during a cycle count, the total amount and the spread of inaccurate locations present in the warehouse. During experimentation these factors are situation dependent. In Chapter 5, we made a simulation model where these factors are adjustable. In Chapter 6, we run multiple experiments to get an understanding of the performance of the proposed method for various situations.

One of the things which we want to investigate during the experiments is the effect of changing the time settings (available counting time, travel time cost or location counting time cost). This can influence the score-cost ratio to visit a location and this can influence the size of the cycle count route, which we expect not to be directly visible by only measuring the FpM. Therefore, we propose the CPI amount of locations counted per cycle count moment CpM and the ratio amount of inaccuracies found per amount counted each cycle count moment ($FpCr$) as given in Equation 4.11.

$$FpCr = \frac{FpM}{CpM} \tag{4.11}$$

Learning performance indicators

The KPI $RMSE$ between the $EFpM$ and the $E[FpM]$ give indications about how well the learning model can predict the amount of inaccuracies it is going to find. This can be measured during both online and offline learning. During offline learning we know the exact OIRA and the chance that a location has the status *inaccurate* ($P(inaccuracy)$) (Equation 5.3). Since we know the exact OIRA, we can also measure the mean absolute deviation (MAD) between the OIRA and POIRA as given in Equation 4.12.

$$MAD(POIRA) = \frac{\sum_{t=0}^{T} |OIRA(t) - POIRA(t)|}{T} \tag{4.12}$$

This MAD gives us insight in how well the prediction model can predict the accuracy of the warehouse. In online learning the exact OIRA is unknown, but it is possible to calculate the POIRA and/or the MOIRA. A comparison between the MAD for the learning model with a MAD for the measurements (Equation 4.13) during offline experiments can give us insight in whether the POIRA or MOIRA will give the best estimate of the OIRA during online learning.

$$MAD(MOIRA) = \frac{\sum_{t=0}^{T} |OIRA(t) - MOIRA(t)|}{T} \tag{4.13}$$

We propose to use one more performance indicator that focuses on the error between individual location prediction value $E[P(inaccuracy)]$ and the actual $P(inaccuracy)$. This is calculated by taking the root of the sum of the individual squared errors between $P(inaccuracy)_i$ and $E[P(inaccuracy)]_i$ and dividing this by the total amount of locations in the warehouse I as given in Equation 4.14 [38].

$$RMSE(P(inaccuracy)) = \sqrt{\frac{\sum_{i=0}^{I} (P(inaccuracy)_i - E[P(inaccuracy)]_i)^2}{I}} \tag{4.14}$$

The RMSE gives the average performance of the individual location predictions in the warehouse, while the MAD (Equation 4.12) gives insight in the learning model's ability to predict the accuracy of the warehouse. In other words, the RMSE analysis the performance of the learning model on macro level (per location), while the MAD focuses on the performance of the learning model on micro level (the warehouse) [26].

Chapter 5

Simulation model

In this chapter, we describe the simulation model, which is used to test our proposed cycle counting method of Chapter 4. We used the Spyder editor to make our simulation model as a Python 3.8 coded model. Our simulation model is based on the data provided by Bolk logistics, described in Chapter 3. The simulation model is a simplified representation of a warehouse that is operated for 24 hours a day and 7 days a week. The simulation model will simulate inventory inaccuracies and transactions between each counting moment. After each counting moment the simulation model processes the counting information of the locations selected by the cycle counting method of Chapter 4.

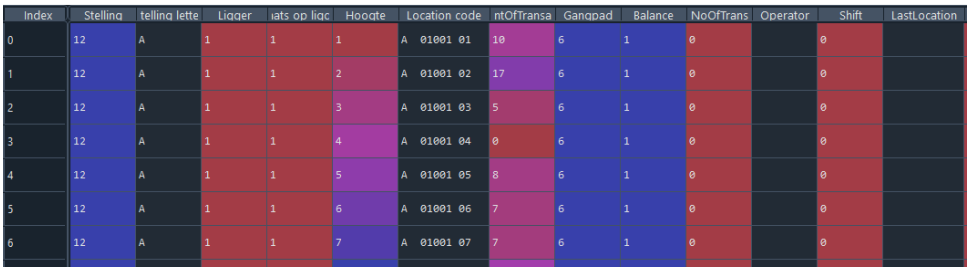
In Section 5.1, we describe the layout of the simulated warehouse and we describe the operations of the simulated warehouse. In Section 5.2 we describe how and when inaccuracies occur in the simulated warehouse in Section 5.3. Thereafter, we describe the use of a warm-up period to initialise the model. In Section 5.4 we explain how the travelling time to visit the selected locations is calculated.

5.1 Simulated warehouse

In Chapter 3, we described the inventory counting and the operations of the warehouse of Bolk logistics. In our simulated warehouse we use Bolk logistics as our use case. We based our layout on the layout of the VNA rack storage at Bolk logistics. We do this such that, we can use the processed historical transaction data from this warehouse, which is described in Section 3.2. It is possible to change the layout of the warehouse as long as the layout is within the dimensions of the warehouse of Bolk (e.g., reducing the amount of aisles). However, in this chapter we only discuss the layout based on the VNA storage of Bolk logistics.

The layout of our simulated warehouse is as follows. We have 6 identical aisles. In each aisle there is a lane of pallet racks on the left hand side and on the right hand side. Each lane of pallet racks has a depth of 20 identical bays. Each bay has a height of 8 shelves, and at each shelf there are 3 pallet locations. This gives us a total of 5760 possible locations for a pallet. Next to the pallet locations, we have one location for the UAV. This location is the start and end point of the UAV for each cycle count. We call this location the docking station and it is placed at the bottom of a lane of pallet racks. In Section 5.4 we describe the routing through this simulated warehouse.

As mentioned before, the warehouse is operated 24 hours a day and 7 days a week. During operations certain pallets are moved from a certain location to another location. In the simulation we only keep track of the movement of the pallets located in the racks. Per location, we keep track of the last transaction that corresponded to the location. Each transaction contains information about multiple parameters. Examples of parameters are, the operator and article number. A snapshot of this table in the Python simulation is given in Figure 5.1.



Index	Stelling	telling lette	Ligger	iats op ligc	Hoogte	Location code	ntOfTransa	Gangpad	Balance	NoOfTrans	Operator	Shift	LastLocation
0	12	A	1	1	1	A 01001 01	10	6	1	0		0	
1	12	A	1	1	2	A 01001 02	17	6	1	0		0	
2	12	A	1	1	3	A 01001 03	5	6	1	0		0	
3	12	A	1	1	4	A 01001 04	0	6	1	0		0	
4	12	A	1	1	5	A 01001 05	8	6	1	0		0	
5	12	A	1	1	6	A 01001 06	7	6	1	0		0	
6	12	A	1	1	7	A 01001 07	7	6	1	0		0	

Figure 5.1: Simulated warehouse: location information table

Depending on the amount of parameters that is kept track of we get a table with the amount of locations as rows and the amount of parameters as columns. Next to the transaction parameters, also information about the location and about previous cycle counts is kept track of in this table. This is information, such as aisle number, rack number, last moment counted and amount of counts. A part of this parameter information is used to generate a chance of failure. How we generate this chance of failure is discussed in Section 5.2. We only want to mention here that all the parameter information is related to the chance of an inaccuracies for each location. In this way the chance of being inaccurate will be different for each location, because we link the transaction and location information to the chance of failure.

5.2 Simulating inaccuracies

In this section we discuss how the inaccuracies in the warehouse are simulated. We want to simulate inaccuracies that are representative to errors that occur in real life situations. Therefore we start this section with an investigation on multiple type of errors that could cause an inaccuracy. Thereafter, we discuss the selection of five parameters and we link them to the found error causes. Finally, we discuss the failure rate function and the calculation of the chance of failure given the selected parameters.

5.2.1 Categories of errors

Kang and Gershwin [25] have categorised the causes of these errors into four categories. These four categories are stock loss, transaction errors, inaccessible inventory, and incorrect product identification.

With stock loss we mean the loss of items that were accurate at the moment it was placed in the warehouse, but somehow got lost over time. Typical examples of stock loss are theft of items, expired shelf life, unauthorised consumption or damaged products [10] [25]. In other words, with stock loss we mean the loss of stock, which we do not detect getting lost and therefore we can not update our inventory record.

Transaction errors are errors that happen during a *transaction*. What we mean with a transaction is each registered movement of an item. Typically, this could be during inbound or outbound shipments or internally in a warehouse. Typical examples of transaction errors are scanning errors, errors in inbound or outbound inventory records [25] [52].

Next to the transaction errors and stock loss there can exist inaccessible inventory. Inaccessible items are items that are present in the warehouse, but are not available because they can not be found [25]. Typical examples of inaccessible inventory are consumers who take a product and place it back at the wrong location, or employees placing products at the wrong location during replenishment [10].

The final category is the incorrect product identification. In other words, the labels on the items are not correct. Scanning these labels will change the inventory level of the wrong products. This could for example happen with inbound products from a manufacturer who used the wrong label [25].

5.2.2 Parameter selection

In our simulation model we make no separation between types of inaccuracies. We only simulate whether the inventory record status of a location is either *accurate* or *inaccurate*. However, we want to link the parameters that we select to at least one of the four categories described above. In our model we select five parameters, but it is possible to increase or decrease the amount of parameters. Increasing the amount of parameters makes the model more complex, which makes it harder for the model to learn the failure behaviour, while decreasing the amount of parameters will make it easier to learn.

The first parameter that we selected is the balance status. We refer to balance as whether or not something is stored at a location. We believe that a location is more prone to theft when there is inventory at a location in comparison to not having inventory at the location. Therefore, we link the chance of having stock loss to the balance status of the inventory record.

The second and third parameter that we select are the operator and shift number. The operator has to execute scanning steps and the movement of pallets during operations. We believe that certain operators make more mistakes than others during operations. Next to that, we believe that the performance of humans is different during different moments of the day, due to for example fatigue. Therefore, we link the different values of the operator and shift number to transaction errors and inaccessible inventory, which both can be caused due to errors during operations.

The fourth and fifth parameter are the height and bay number of the pallet location. We believe that placing a pallet at certain heights and certain bays is more difficult than other, which can increase the chance of a damaged product in comparison to a better accessible location. Further in this section we refer to these parameters as a specific parameter number u, where each value represents a parameter as given below.

- $u = 1$ = Balance
- $u = 2$ = Operator
- $u = 3$ = Shift number
- $u = 4$ = Height
- $u = 5$ = Bay number

5.2.3 Inaccuracy function

We now discuss the location failure rate function. We define the location failure rate λ as the amount of failures that occur per day (24 hours) at each location. We assume that failures at each location occur at random moments over time and that the failure rate depend on u individual parameters u. Each parameter u has its own failure rate λ_{uv} for each parameter value v. Here, we assume that failures follow an exponential failure distribution [21]. By assuming that the failure rate of a location is exponentially distributed we can sum the individual parameter value failure rates λ_{uv}, such that we get the location failure rate λ as given in Equation 5.1.

$$\lambda = \lambda_{1v} + \lambda_{2v} + \lambda_{3v} ... + \lambda_{uv} \tag{5.1}$$

As mentioned in Section 5.2.2, we select five parameters u. However, it is possible to increase or decrease the number of parameters for the failure rate function. From historic failure data it is possible to calculate an estimate of λ_{uv} for each value v of these five parameters u [22]. This can be done by dividing the sum of the failures that belong to parameter value v by the amount of days. This would give us the individual parameter value failure rate λ_{uv}. Nevertheless, we expect that assigning a failure to a parameter value will be a challenge.

We do not have failure data available. Next to that, we want to be able to adjust the failure rate for various simulations. Especially, we want to be able to increase or decrease the effect of a certain parameter u. By splitting the parameter value failure rate λ_{uv} into a parameter weight a_u and a parameter value variable x_{uv}, we do not have to adjust each λ_{uv} to chance the effect of a parameter. Here, the weight a_u represents the maximum amount a parameter u can contribute to the location failure rate and x_{uv} represents how much is contributed based on parameter value v. We divide the contribution of

a parameter value into *high*, *medium*, and *low*. We assign value 1 to label *high* and a value between 0 and 1 is assigned to label *medium* and *low*, where the value for label *medium* is bigger than for label *low*. Therefore, we can also write Equation 5.1 as Equation 5.2.

$$\lambda = a_1 * x_{1v} + a_2 * x_{2v} + a_3 * x_{3v} ... + a_u * x_{uv} \tag{5.2}$$

We define the sum of the weights a_u over every parameter u as the maximum possible failure rate. This is the case when the contribution of each parameter value x_{uv} of a location is labelled *high*. As mentioned before, the weight a_u represents the maximum effect of a certain parameter to the failure rate and it depends on the contribution of a parameter value how strong this effect is. To illustrate this we provide an example for shift number. The shift number has possible values $v = 1$, $v = 2$, and $v = 3$ and we set the weight value to 0.05. We assign for shift number 1 a contribution value of 1 (label *high*), shift number 2 a contribution score of 0.1 (*low*) and shift 3 a contribution score of 0.5 (*medium*). Then the contribution to the failure rate for $v = 1$ is equal to 0.05, for $v = 2$ the contribution is 0.005, and for $v = 3$ it is equal to 0.025. As can be seen the effect of the parameter shift number to the failure rate is 10 times as high for $v = 1$ in comparison with $v = 2$. In this way it is also possible that parameters with a lower weight a_u can have a higher contribution to the failure rate depending on the parameter value score x_{uv}.

Above, we described the calculation of the individual location failure rate λ. In our simulation model the value of λ depends on the parameter values v at time moment t. It depends on time moment t, because after each count moment t we simulate transactions and those transactions influence the parameter values. When each individual value of $\lambda(t)$ is known, we can calculate the chance of an inaccuracy over a certain time period [33]. When we set $t = t_{begin}$ as the start moment of the time period and $t = t_{end}$ as the end of the period, then we can calculate the chance of an inaccuracy ($P(inaccuracy)$) as given in Equation 5.3. This equation represents the chance that at least one inaccuracy occurs during the period. Where the product of $(1 - \lambda(t))$ over each period t, represents the chance that no inaccuracy occurs during these periods.

$$P(inaccuracy) = 1 - (1 - \lambda(t_{begin})) \times (1 - \lambda(t_{begin} + 1)) \times ... \times (1 - \lambda(t_{end} - 1)) \times (1 - \lambda(t_{end})) \tag{5.3}$$

In our model, we get for each individual location a failure rate $\lambda(t)$ at time t. Each count moment t, we draw for each location a random number between 0 and 1. When the random number is less or equal to the failure rate, then we set the location inventory record status to *inaccurate*. As mentioned in Chapter 4 the record status stays *inaccurate* until it is visited during a cycle count. After a visit the location record status becomes *accurate* and one time period t later the chance of an inaccuracy is equal to the equation given in Equation 5.3. We want to note here, that the NN of Step 2 Section 4.4 tries to predict this $P(inaccuracy)$ for each location. Therefore, next to the five selected parameters of Section 5.2.2, the NN also considers last count moment per location.

5.3 Warm-up period

We are simulating inaccuracies in a warehouse. At the beginning there are no inaccuracies. With the combination of the transaction data and the inaccuracy function, we want to simulate those inaccuracies such that there is a correlation between the parameters of the location (transaction) data, and being accurate or inaccurate.

So, the inaccuracies occur when we let our model run. There should be a representative amount of inaccuracies in our simulated warehouse before we let our learning model train. Otherwise our learning model will only encounter accurate records for the first couple of runs. This is not desirable since our learning model will think that all parameters encountered in the beginning will be good parameters to estimate that a record is accurate. While it can be that we simulate those parameters as indicators for being an inaccuracy.

When inaccuracies occur and no corrections are made, the model's OIRA drops. By setting a threshold for the OIRA, for example 90%, we can determine a warm-up period. A warm-up period is a period where the model runs, but we do not collect any information during this period to process [47]. The idea is that we let the model run, until desired conditions are met by the model. In our case we define the

warm-up period as the period until we reach our desired starting OIRA.

When we simulate 365 days and each period t has a length of 1 day, then we have 365 periods t. Then we can determine our warm-up period by starting at period $t = 0$, where we do not execute cycle counting but we simulate transactions and inaccuracies. After each period t the OIRA declines. We determine the warm-up period as the length of the time between $t = 0$ and the first period where the OIRA is equal to or lower than our threshold. In Figure 5.2 it is illustrated how the warm-up period is determined. Every period t on the left side of the vertical dashed line is part of the warm-up period and every period on the right side of the vertical dashed line we use to test our cycle counting method on.

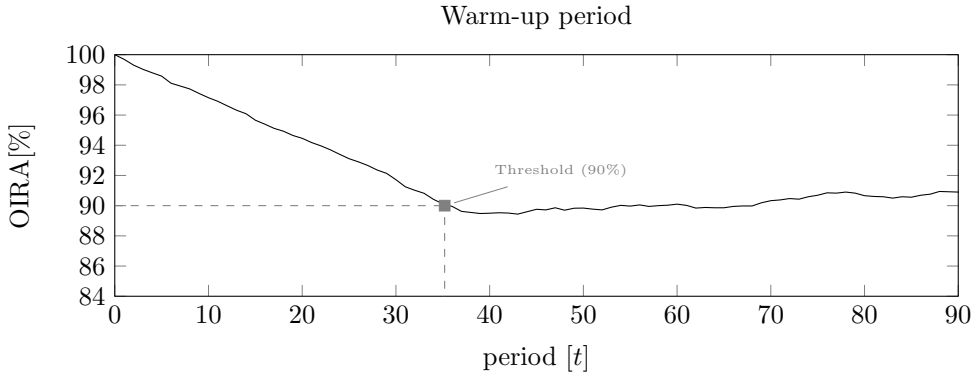


Figure 5.2: Example: Determine warm-up period

5.4 Distance model

In Section 4.3, we already discussed the time restrictions per cycle count moment. Here we recap these restrictions and we further elaborate on these restrictions. The amount of locations (n) that an UAV can visit during a cycle count, depends on the following four factors:

1. The available counting time (ACT)
2. The traveling time between selected locations (y_{ij})
3. The scanning time per location ($STPL$)
4. Battery capacity of UAV (between charges)

In this research we are seeking for a sample selection policy that could perform under time restrictions. These time restrictions are counting intervals of 15 minutes or less. From benchmarking publically accessible information from two other companies (Ware [48] and Arox [50]) that use UAVs for inventory counting, we found that the battery capacity of 15 to 20 minutes is used. Therefore, we assume that the battery life of the UAV should not be a restricting factor. So, the amount of locations n we could select for our sample depends on the factors 1, 2, and 3. We consider ACT and $STPL$ as fixed input parameters and are not dependent on which locations are selected. Nevertheless, The Y_{ij} is dependent on the set of selected locations (X_{ij}). The relation between the ACT, the Y_{ij}, the amount of locations selected n, and the $STPL$ is presented in constraint 4.5 of the mathematical model of Section 4.3.

The TTBSL is dependent on the selected locations. We will use a model that determines the total travelling distance between the selected locations and from and back to the docking station. The total distance travelled is split into horizontal movement and vertical movement. We will first start by explaining consecutively the horizontal movement calculations, the vertical movement calculations and combined calculations. Thereafter, we will provide a few simple examples. For these examples, step by step calculations can be found in Appendix B. Finally, we will use these examples to verify the distance calculations in our Python model.

Horizontal movement

We assume that there are an given amount of Aisles. As shown in Figure 5.3, each aisle contains two lanes of pallet racks (one on the left and one on the right). Next to that, each aisle has one entry point

at the bottom. To go to the next aisle the UAV has to use the bottom entrance and could only exit via this bottom entrance. In other words, it is only possible to go from one aisle to another aisle via the bottom, an UAV could not fly over the racks or fly through an empty shelf. The UAV always starts and finishes its cycle count at the docking station. Therefore, we include this distance in our calculation. We calculate the total horizontal movement as follows:

1. Start at the docking station (DS)
2. Determine the most left aisle, which need to be visited
3. Determine distance from docking station DS to most left aisle
4. For each aisle determine the furthest pallet location that need to be visited (*This is the location with the highest bay B and location on shelf LoS number on that bay*)
5. Determine the horizontal distance to be travelled per aisle. (*Two times the distance to the furthest pallet location (back and forth)*)
6. Calculate the total sum over all the horizontal distances travelled per aisle
7. Determine most right aisle that need to be visited
8. Determine distance most left aisle to most right aisle
9. Determine distance most right aisle to DS
10. Calculate the total horizontal distance travelled. (*The sum over the distances determined at 3, 6, 8, and 9*)

In our simulated warehouse, there are 6 VNA aisles (Ai), with 12 lanes of pallet racks ($LoRn$). Each lane of pallet racks has a depth of 20 bays (B). Each bay has a height (h) of 8 shelves and at each shelf there are 3 pallet locations (LoS). In other words, we could say that the horizontal depth per aisle is equal to 60 pallets. As stated in the introduction of this section, we will provide some example calculations later in this section, but first we will explain the vertical movement.

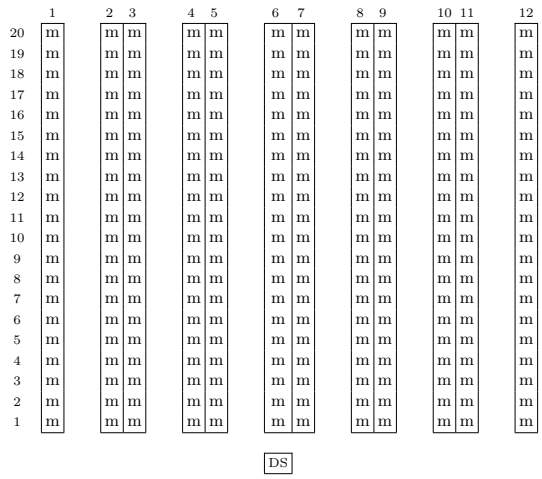


Figure 5.3: Distance calculation model: Amount of aisles = 6, amount lanes of pallet racks = 12, amount of bays per rack = 20, height (amount of shelfs per bay) = 8, Locations per shelf = 3, Docking station location (Pallet rack lane number) = 6

Vertical movement

Next to the horizontal movement through the aisles, the UAV also has to fly in vertical direction. The displacement in height for the UAV is neccessary to reach the shelf on which the pallet is placed. In Figure 5.4 a front view of one of the twelve racks of Figure 5.3 is given. We calculate the total vertical movement in terms of difference in shelf height. We do this as following:

1. Start with starting height 1 (height of docking station)
2. Retrieve height of first location of the horizontal movement route. (*Rule of thumb: this is the location with the lowest Aisle number and lowest bay number for that aisle (if there are multiple then this is sorted based on location on shelf and if necessary on the lowest height))*
3. Calculate absolute height difference between start and first location
4. Retrieve next location of horizontal movement route and check if aisle number is equal to last location

   If aisle number is equal:

   - Calculate absolute height difference between retrieved and last location.

   Else:

   - Calculate absolute height difference between docking station and retrieved location
   - Calculate absolute height difference between previous location and docking station.
5. Repeat step 4 until last item is reached
6. Calculate absolute height difference between last item and docking station
7. The horizontal movement is the sum of all of the absolute height differences

As already stated in Section 5.4, there are six VNA aisles with pallet racks. Each of these racks have a height of 8 shelves / 12.5 metres in total. We assume that we will fly at shelf height to scan the pallet location. So the maximum difference between two consecutive scanning locations is 7 ($Shelf8 - Shelf1$).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
8	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m
7	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m
6	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m
5	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m
4	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m
3	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m
2	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m
1	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m

Figure 5.4: Distance calculation model (vertical representation): amount of bays = 20, amount of locations per shelf = 3, height per bay = 8, docking station height = 1

Total movement: combining vertical and horizontal movement

The total distance an UAV has to fly is the combination of vertical and horizontal movement. We expressed the horizontal movement in terms of pallet locations and the vertical movement in terms of shelf height. In case of Bolk logistics, these distances are in practice not equal. The width of a shelf of Bolk logistics is around 2.70m on which three euro-pallets fit. The horizontal distance between consecutive pallet locations is around 90cm. The height of Bolk's bays are around 12.5m, which makes the distance between each of the eight shelves around 156cm.

We want to provide a general model and not a Bolk specific model. Therefore, in our calculation examples we say that a unit of vertical movement is equal to a unit of horizontal movement. However, it is possible to adjust this ratio by multiplying the horizontal and vertical movement with a desired ratio (e.g., for Bolk 0.90 and 1.56). When the ratios are equal, visiting a location directly on the left or the right of a certain location takes the same travelling time as visiting a location directly above or beneath that certain location. Next to that, we assume that it is not possible for an UAV to fly diagonal. However, in practice an UAV might be able to fly diagonal. We do not use diagonal movement since we only want to approximate the total movement and it is not the main scope of this research to optimise the routing. Therefore, we calculate the total movement as the sum of the vertical movement and the horizontal movement.

Calculation examples

For each of our examples we place the docking station DS at lane of racks number $LoRn = 6$.

Example 1

In our first example we visit 4 locations all at height $h = 1$ and location on shelf $LoS = 3$. All 4 locations are in Aisle 1. The corresponding route is presented in Figure 5.5. When we use the horizontal and vertical movement rules described above, then we find a total movement of 122 for example 1. An extensive calculation can be found in Appendix B.1.

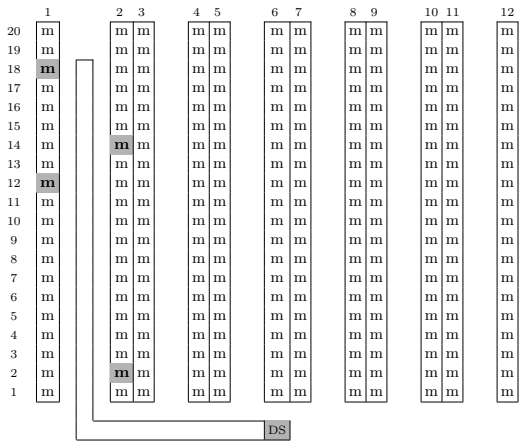


Figure 5.5: Example 1: 4 locations visited in most left aisle

Example 2

In example 2 we visit 8 locations divided over 2 different aisles. In this example we keep the height for each location equal to 1 and location on the shelf equal to 3. Four of the locations are placed in Aisle 1 and four are placed in Aisle 6. The corresponding route is presented in Figure 5.6. Applying the logic above we find a total movement of 210. Extensive calculations can be found in Appendix B.2.

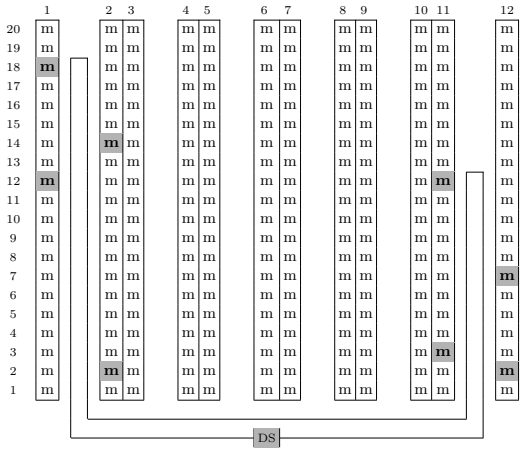


Figure 5.6: Example 2: 8 locations visited divided over aisle 1 and aisle 6

Example 3

In example 3 we visit 15 locations divided over all 6 aisles. In this example we keep height 1 and location on shelf = 3. The corresponding route and locations is presented in Figure 5.7. The total movement in this example is equal to 528. Extensive calculations are given in Appendix B.3.

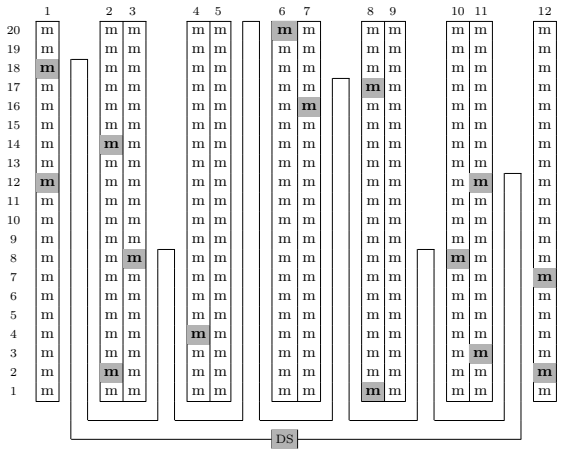


Figure 5.7: Example 3: 15 locations visited, divided over all six aisles

Example 4

We take our first example where we visited 4 locations in the same aisle (Figure 5.5. We keep the horizontal movement route, but we adjust the height of the locations. The vertical route is presented in Figure 5.8. (Note: we keep the location on shelf = 3). The total movement of this example including vertical movement is equal to 142. In Appendix B.4 we provide the step-by-step calculations for this example.

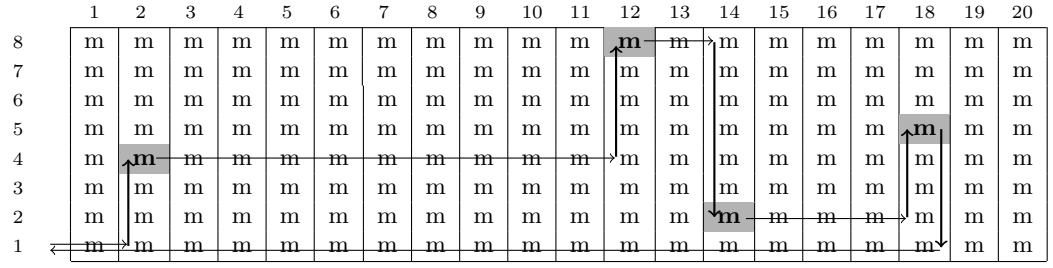


Figure 5.8: Distance calculation model: Amount of aisles $Ai = 12$, amount of racks per aisle $RpA = 20$, Docking station location $DS = 6$

Verification of implementation in Python model

The above given examples are used to verify the distance calculations in our simulated warehouse model. Before we can do this we have to translate the selected sample locations of our examples to *locationcode*. This is necessary, because our model uses the *locationcode* as an unique identifier to process all transactions and cycle count information. In Table 5.1 an explanation is shown how to construct a location code based on rack, bay, place on shelf, and height as inputs. Since the UAV will fly through aisles and not through racks we also have to assign racks to aisle numbers. This is presented in Table 5.2.

Location code	A xx00y 0z					
Symbol	A	xx	00	y	0	z
Definition	Rack	Bay		Place on shelf		Height
Low	L(=1)	1		1		1
High	A(=12)	20		3		8

Table 5.1: Explanation of location code

Aisle	Rack
6	A,B
5	C,D
4	E, F
3	G, H
2	I, J
1	K, L

Table 5.2: Link aisles to racks

In Appendix B, we provide for each example the results given by our coded model. As can be seen in this appendix, the results given by the coded model are equal to the results given by our manual calculations. Therefore, we further assume that our distance model is correctly implemented in our coded model.

Chapter 6

Experiments

In this chapter, we present the experiment results of the proposed cycle counting method of Chapter 4. We executed five types of experiments in our simulated environment from Chapter 5. In the simulation model it is possible to adjust multiple settings such as the size of the warehouse. In Section 6.1, we discuss the main experiment settings and at the end of the section we propose four base models. Every experiment the main settings are equal to the selected base model unless explicitly mentioned otherwise. In Section 6.2 we present per experiment type which base models are used and we present per experiment which specific adjustments are made to the settings of the base model.

In Section 6.3, we present the results of the experiments described in Section 6.2, where each type of experiments consist of multiple experiments. The five types of experiments researches the following:

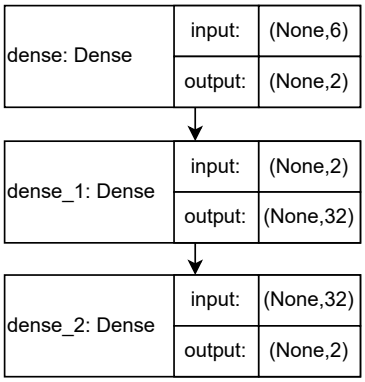
1. The effect of the seed (first selected) location of each cycle count moment
2. The performance of the learning model using the neural network in comparison to two other regression based learning methods.
3. The effect of adding a learning period, using the proposed exploration and exploitation balance methods, on the (learning) performance of the algorithm.
4. The effect of the length of phase I (learning period) to the overall and learning performance.
5. Sensitivity analysis: the effect of parameter settings on the performance of the proposed method in comparison to a traditional cylce counting method.

6.1 Experiment settings

In this section we first discuss the general settings. These are settings of the learning model (6.1.1), the failure rate (6.1.2), the warehouse layouts (6.1.3) and time settings (6.1.4). For the learning model and the time settings we use the same settings for each experiment unless explicitly stated otherwise in the experiment description. For the warehouse layout we constructed three different environments and for the failure rate we made two different settings. From this we constructed four base models in Section 6.1.5. Thereafter, in Section 6.2 we discuss per experiment type which experiments we conduct using which base model and the experiment specific settings.

6.1.1 Neural network settings

For each of the base models of Section 6.1.5 we use the same learning model. This learning model is a neural network (NN). We used the Python library Keras to built our NN [44]. This NN model consist of an input layer, two hidden layers and one output layer. A summary of our model is shown in Figure 6.1. In Keras our NN model is called a "Sequential" model and the type of layers that we used is called a "Dense". The output shape of a Dense layer is equal to the amount of neurons of that layer. For both hidden layers, we used the Sigmoid activation function. For the output layer of the NN we use the Softmax activation function. These activation functions are selected after some tests with a small test and training dataset. During these tests the most common used activation functions for the hidden layers and output layers are compared [13].



(a) NN: connection of layers

Layer	Type	Output Shape	Param #	Activation
Input		(none,6)		
Hidden	Dense	(none,2)	14	Sigmoid
Hidden	Dense	(none,32)	96	Sigmoid
Output	Dense	(none,2)	66	Softmax
Total parameters: 178				
Trainable parameters: 178				

(b) Sequential neural network model

Figure 6.1: Neural network basic model

During the compile stage of the NN one should select a loss function and an optimiser function. These are selected by running some tests (with the same dataset as for selecting the activation functions) with often used loss and optimiser functions [37]. The selected loss function is the Sparse Categorical Crossentropy and the optimiser function is the Adam optimizer with learning rate 0.01 [12].

After each cycle count the model is fit to cycle data gathered during previous cycle counts. However, first the data is shuffled and split into a train and test data set (80:20). This split into a train and test data set is executed to calculate a validation loss value during the fitting of the model [14]. During the fitting of the model to the data a callback function "Early stopping" monitors this validation loss. This callback function prevents the model to overfit to the train data. During fitting the train data is split into multiple batches of a predefined batch size. In our model we set the batch size to 480. For example, we have a training set consist of 1920 entries, then we get four different training batches. The amount of times we go over these batches is defined as the amount of epochs. We set the amount of epochs to 500, this means that the model is fitted at most 500 times over each of the four batches. However, after each epoch the validation loss is calculated. If a model is over fitted to the train data during consecutive epochs, the validation loss will increase. The callback function "Early stopping" keeps track of how often the validation loss increases after consecutive epochs. The callback function has a certain threshold "Patience", which defines how often the validation loss of two consecutive epochs is allowed to increase. If this threshold is met, the callback function will stop the fitting of the model and will restore the best weights for the channels of the NN that correspond to the epoch with the lowest validation loss (Restore_best_weights=True). The code of the implementation of this NN model, for making the predictions $E[P(inaccuracy)]$, is given in Appendix C.

6.1.2 Failure rate settings

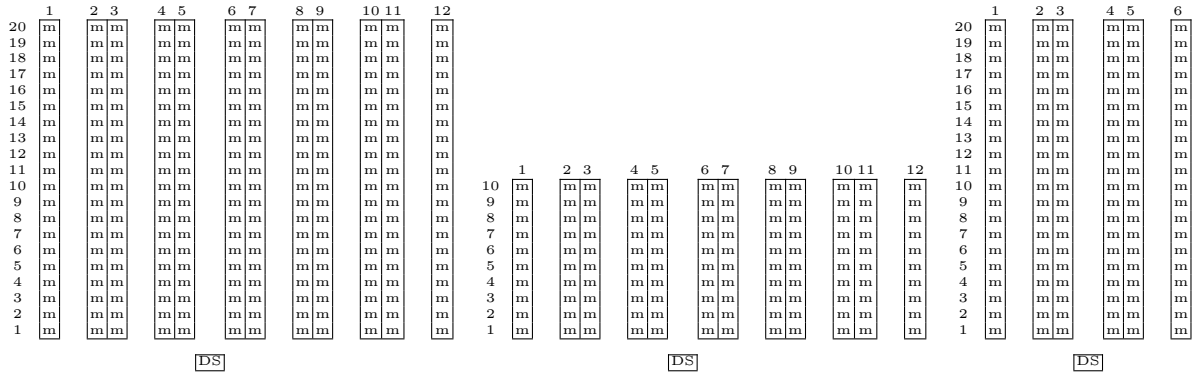
During the experiments we compare two different settings for the failure rate. As discussed in Section 5.2, the failure rate λ_i is equal to the sum of the individual parameter value failure rate λ_{uv} (Equation 5.1). Each λ_{uv}, consist of a weight a_u and a parameter value variable x_{uv}. As mentioned, in Section 5.2, the effect of x_{uv} is either *high*, *medium*, or *low*. In table 6.1, we present the parameter weights a_u for the two failure rates. Next to that, we present the value for the label *low*, *medium*, and *low*. The difference between failure rate setting 1 and 2 is that for setting 1 the effect of each parameter u is equal if the parameter value v has the same label, while for setting 2 this is not the case. Next to that, the difference between values for the status *low*, *medium*, and *high* are much higher for setting 2.

	a_1	a_2	a_3	a_4	a_5	*Low*	*Medium*	*High*
Setting 1	0.001	0.001	0.001	0.001	0.001	0.05	0.5	1
Setting 2	0.0001	0.00005	0.00005	0.00515	0.00515	0.0005	0.01	1

Table 6.1: Two type of failure rate settings

6.1.3 Physical environments

During the experiments we want to compare different warehouse layouts. Here we describe the three warehouse environments which we used. First, we briefly describe the physical dimensions and characteristics of each environment. Thereafter we present the dimensions of each environment in Table 6.2. A physical representation of the layout of the environments is given in Figure 6.2.



(a) Environment 1 (b) Environment 2 (c) Environment 3

Figure 6.2: Layout of environments

Environment 1

Environment 1 is based on the size of the VNA storage of Bolk logistics. It is the largest of the three environments. It has six long identical aisles, entry and exit is only possible via the bottom of each aisle. The docking station is placed at the bottom of the third aisle.

Environment 2

Environment 2 is smaller in comparison to environment 1. It has the same amount of aisles as environment 1, but the length of each aisle of environment 2 is half the length of the aisle length of environment 1. The docking station is placed at the bottom of the third aisle.

Environment 3

Environment 3 is also smaller in comparison to environment 1. The amount of locations is equal to the amount of locations from environment 2. However, the length of each aisle in environment 3 is the same as in environment 1, but the amount of aisles is reduced from six to three aisles. The docking station is placed at the bottom of the second aisle.

	Environment 1	Environment 2	Environment 3
No. of aisles	6	6	3
No. of lane of racks	12	12	6
No. of bays	20	10	20
No. of shelves per bay	8	8	8
No. of locations per shelf	3	3	3
Total amount of locations	5760	2880	2880

Table 6.2: Dimensions of experimental environments

6.1.4 Time settings

During the experiments we use the same time settings for each of the environments. The time settings are the scanning time per location (STPL), the travel time factor, the available counting time (ACT) and the threshold value for the warm-up period. The travel time factor is the amount of time that it takes to execute one unit of calculated distance (5.4). The time settings are shown in Table 6.3

Setting	Value
Scanning time	6s
Flying time factor	1
Available counting time	900s
Start OIRA level	90%

Table 6.3: Settings of experimental environments

6.1.5 Base model settings

we conducted experiments using four base models (A,B,C,D). Each of the base models use the same neural network and time settings. Base model A and B have the same layout (environment 1), but their simulated failure rate is different (1 and 2). Base model C and D have the same failure rate settings (1) as base model A, but the warehouse size (amount of locations) of model C and D are half the size of model A. The only difference between base model C and D is the shape of the warehouse (environment 2 and 3). An overview of the base model settings is given in 6.4

Base model	Environment	Failure rate
A	1	1
B	1	2
C	2	1
D	3	1

Table 6.4: Base model settings

6.2 Individual experiment settings

In this section we discuss the individual experiment settings per experiment type. Per experiment type it is mentioned which base model is used and which experiment specific settings are changed.

6.2.1 Seed customer

At each count moment the UAV starts at the docking station (DS). As mentioned in Step 3 of Section 4.4, the first selected location has a high effect on which locations are likely to be selected next. Having a first location further away from the DS can overcome that only routes close to the DS are constructed. Since, after selecting the seed customer the algorythm selects a location for a count moment based on the ratio *score* divided by the *additional location counting cost*.

In step 3 of our proposed inventory cycle counting method, we propose five different rules for selecting a seed customer. We test each of these five seed customer rules in each base model. Next to that, we compare these five seed customer rules to selecting the seed customer the same way as locations are added to the route in Step 5 (Seed customer rule 6). This results in 24 experiments, as shown in Table 6.5. The goal of these experiments is to find which seed customer rule gives the best result in terms of OIRA performance.

Exp No.	Model	Seed rule	Exp No.	Model	Seed rule	Exp No.	Model	Seed rule
1	A	3	9	B	5	17	C	2
2	A	4	10	B	1	18	C	6
3	A	5	11	B	2	19	D	3
4	A	1	12	B	6	20	D	4
5	A	2	13	C	3	21	D	5
6	A	6	14	C	4	22	D	1
7	B	3	15	C	5	23	D	2
8	B	4	16	C	1	24	D	6

Table 6.5: List of experiments: seed customer rule

6.2.2 Benchmark learning models

During the second type of experiments we want to get an insight in how well our NN performs in comparison to two other learning methods. We compare the neural network (NN) to two other methods, which can predict the chance of being inaccurate. These two methods are linear regression (LiRe) and logistic regression (LoRe). We do this by swapping the NN model for the other learning methods. In Table 6.6 an overview of the experiment settings are shown. The linear regression and logistic regression models are built using the Python library Keras and are based on [6] and [1]. Our goal is to get an insight of the performance of the NN in comparison to two other regression methods in terms of OIRA performance and the prediction performance. The code for the linear regression and the logistic regression model configuration are shown in Appendix

Exp No.	Base model	Score method	Exp No.	Base model	Score method
25	A	Neural network	28	B	Neural network
26	A	Linear regression	29	B	Linear regression
27	A	Logistic regression	30	B	Logistic regression

Table 6.6: List of experiments: benchmarking learning methods

6.2.3 Introduction of phase I

The third type of experiments compares the use of three exploration-exploitation techniques during phase I. During these experiments we set the length of phase I to 90 days (after the warm-up period). The learning techniques are mentioned in Section 4.5.3. The three techniques that are selected are UCB-1, ϵ-greedy, and Thompson sampling. The UCB-1 technique focuses on the amount of times a location is counted. The UCB-1 score consist of a prediction value of the NN and ads an upper bound value based on the amount a location is counted n_i at cycle count moment t ($\sqrt{2\log(t)/n_j}$). The ϵ-greedy technique splits the cycle count into an exploration part and an exploitation part. After adding a location to the cycle count route the remaining time of the ACT reduces. We refer to the time that is available after adding n locations to the route as remaining ACT ($R(ACT)_n$), where $R(ACT)_n$ is equal to ACT for $n = 0$. The ϵ-greedy technique start with exploration part ($R(ACT)_n \geq (1-\epsilon) \times ACT$) and shifts to the exploitation part after adding a certain amount of locations ($R(ACT) \geq (1-\epsilon) \times ACT$). During exploration the technique assigns a random score value to each location and during the exploitation part the score values per location are given by the NN. The last technique focuses on the outcome of historical cycle counts. The Thompson sampling method calculates a score by using the beta distribution where the times the location status was found accurate (β) and the times it was found inaccurate (α) is used as input. In Table 6.7 we present the experiment specific settings.

Exp No.	Base model	Score method	Exp No.	Base model	Score method
31	A	UCB-1	37	B	UCB-1
32	A	Thompson	38	B	Thompson
33	A	ϵ-greedy ($\epsilon = 0.3$)	39	B	ϵ-greedy ($\epsilon = 0.3$)
34	A	ϵ-greedy ($\epsilon = 0.6$)	40	B	ϵ-greedy ($\epsilon = 0.6$)
35	A	ϵ-greedy ($\epsilon = 1$)	41	B	ϵ-greedy ($\epsilon = 1$)
36	A	No phase I	42	B	No phase I

Table 6.7: List of experiments: introduction of phase I

6.2.4 Length of phase I

After selecting the best technique for phase I, we want to get insights on the effect of the length of phase I. Therefore, we select base model A and B, which both have different failure rate settings. The experiment specific settings are shown in Table 6.8.

Exp No.	Base model	Length phase I	Exp No.	Base model	Length phase I
43	A	0	50	B	0
44	A	30	51	B	30
45	A	60	52	B	60
46	A	90	53	B	90
47	A	120	54	B	120
48	A	150	55	B	150
49	A	180	56	B	180

Table 6.8: List of experiments: length of phase I

6.2.5 Sensitivity analysis

During the last set of experiments we compare the performance of the proposed cycle counting method to two traditional cycle counting methods, which we call the benchmark policies. The first policy tries to eliminate as much as possible inaccuracies, by visiting as much as possible locations per cycle count. The second policy focuses on the quality of MOIRA, by taking random samples from the warehouse. During the experiments we change the settings of some of the adjustable parameters. The parameters which are changed, have the most effect on the performance of the cycle counting methods according to our believe. Below, we first discuss the benchmark polices, thereafter we discuss per parameter which experiments we conduct. We only use the second type of benchmark policies for the last experiments to compare the POIRA and MOIRA of our proposed method to the MOIRA of the two benchmark policies.

Benchmark policies

The first benchmark policy can be compared to the location based cycle counting method described in Section 2.1 and is also referred to as progress-based cycle counting [4]. This cycle counting method starts at the most left aisle and counts every consecutive location in this aisle until every location in the aisle is counted. After counting each location in an aisle, the next aisle is counted until every location in the warehouse is counted. After counting every location in the warehouse, the UAV starts counting again at the most left aisle. In the list of experiments we refer to this benchmark policy with the letter B and for our proposed cycle counting method we assign the letter P.

The second benchmark policy can be compared to random sampling. In Chapter 2, we found that random sampling is often used for estimating the overall warehouse accuracy. The second benchmark policy takes a randomly selected set of locations from the list of VNA locations. It does not consider the distances between consecutive locations. Every round each location has the same chance of being selected for the cycle count. We refer to this random sampling benchmark policy with the letter R. We only use this second type of benchmark policies for the last experiments to compare the performance of our proposed method in terms of estimating the warehouse accuracy.

Scanning time and flying time factor

Locations are added to a cycle counting route based on the highest score-cost ratio. The cost factor of this ratio consist of the fixed scanning time and the additional time to visit the location. By either adjusting the fixed scanning time or the flying time factor, the cost factor of the ratio changes. We want to get an insight in whether the model performs better or worse in comparison to the benchmark policy when adjusting these values. The exact experiment settings are shown in Table 6.9

Exp No.	Base model	Scan time	Flying time factor	Method
57	A	3	1	P
58	A	3	1	B
59	A	6	1	P
60	A	6	1	B
61	A	12	1	P
62	A	12	1	B
63	A	6	0.5	P
64	A	6	0.5	B
65	A	6	1	P
66	A	6	1	B
67	A	6	2	P
68	A	6	2	B

Table 6.9: Sensitivity analysis: scan time and flying time factor settings

Available counting time

The available counting time influences the total amount of locations that can be visited during each cycle count. We want to get an insight in whether the model performs better or worse in comparison to the benchmark policy when the amount of location counted per count moment increases or decreases. The exact experiment settings are shown in Table 6.10.

Exp No.	Base model	ACT	Method	Exp No.	Base model	ACT	Method
69	A	10	P	72	A	15	B
70	A	10	B	73	A	20	P
71	A	15	P	74	A	20	B

Table 6.10: Sensitivity analysis: available counting time settings

Failure rate weights

The height of the failure rate weights influence the total amount of failures that occur per cycle count moment. As mentioned in 1.4, the direction of the OIRA depends on the balance between inaccurate records found and the amount of new inaccurate that occur. We want to get an insight in whether the model performs better or worse in comparison to the benchmark policy when the amount of failures that occur per cycle count moment increases or decreases. The exact experiment settings are shown in Table 6.11.

Exp No.	Base model	a_u	Method	Exp No.	Base model	a_u	Method
75	A	0.0005	P	78	A	0.001	B
76	A	0.0005	B	79	A	0.002	P
77	A	0.001	P	80	A	0.002	B

Table 6.11: Sensitivity analysis: failure rate weight settings

Layout

During this set of experiments we want to get an insight in whether the shape of the warehouse influences the performance of the proposed model in comparison to the benchmark model. Next to that, we run two experiments per base model to compare the effect with and without the phase I settings determined during the third and fourth type of experiments. The exact experiment settings are shown in Table 6.12. The value T depends on the determined length for phase I.

Exp No.	Base model	Len. phase I	Method	Exp No.	Base model	Len. phase I	Method
81	C	0	P	84	D	0	P
82	C	T	P	85	D	T	P
83	C	0	B	86	D	0	B

Table 6.12: Sensitivity analysis: layout settings

OIRA estimations

During the last experiments we want to get an insight in how well the proposed model can estimate the actual OIRA in comparison to both benchmark policies (B & R). For each of these models we want to compare the monthly moving average of the MOIRA and for the proposed model also the monthly moving average POIRA, to the actual OIRA. The exact experiment settings are shown in Table 6.13.

Exp No.	Base model	Method	Exp No.	Base model	Method
87	A	P	90	B	P
88	A	B	91	B	B
89	A	R	92	B	R

Table 6.13: Sensitivity analysis: failure rate weight settings

6.3 Experimental results

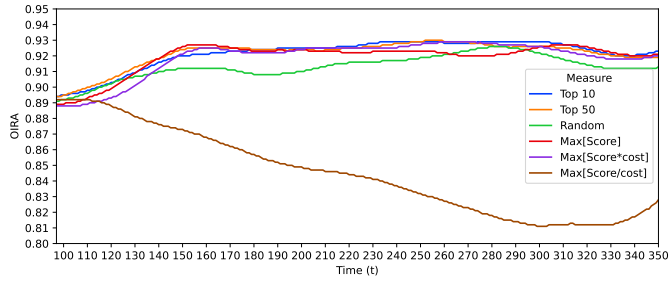
In this section we present the results of each of the five different types of experiments mentioned at the begin of this chapter. For each experiment we present the OIRA result in a graph. In these graphs, we present the OIRA in terms of the monthly moving average of the OIRA (MMA(OIRA)). We use the MMA to smooth the curves and filter out the noise, such that we see the general development of the OIRA over time [35]. This also makes it easier to compare experiments among each other. Next to the graphs we present tables with (key) performance indicators (Section 4.6). These performance indicators give an indication how well the model performs on certain aspects. In these tables abbreviations are used for the performance indicators. In Table 6.14 an explanation is given per abbreviation. The OIRA graph and the corresponding performance indicators table have the same cycle counting time range (e.g., range 100-350).

Abbreviation	Explanation
Exp.	Experiment number
Found	Total amount of inaccuracies found during the whole cycle counting horizon.
Predicted	Total amount of inaccuracies predicted to find during the whole cycle counting horizon.
Count	Total amount of locations counted during the whole cycle counting horizon.
Found/count	Ratio amount of inaccuracies found per counted location
$\overline{OIRA}$	Average OIRA over the whole cycle counting horizon
$\overline{POIRA}$	Average POIRA over the whole cycle counting horizon
MAD	Average of the individual absolute deviations between OIRA and POIRA (Equation 4.12)
$\overline{RE}$	Average of the individual relative errors between inaccuracies found and predicted to find (Equation 4.10)
$\overline{RMSE}$	Average of the individual RMSE per count moment (Equation 4.14)

Table 6.14: Abbreviations used in performance indicator tables

6.3.1 Seed customer

Here, we present the results for the experiments regarding seed customer rules. In consecutive, Figure 6.3, Figure 6.4, Figure 6.5, and Figure 6.6 we present the results for the experiments conducted with Base Model A, Base Model B, Base Model C, and Base Model D. In each of the figures, the MMA(OIRA) is shown (a), and a table with performance indicator results (b). The performance indicators are calculated over the whole length of the cycle counting period. This counting period starts after the warm-up period up to and including counting day 365.

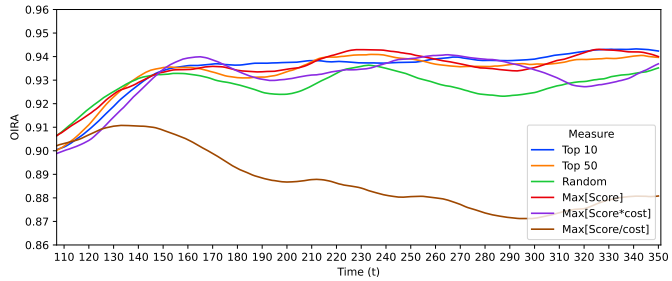


(a)

Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
1	4389	4524	30624	0.143	0.920
2	4342	4366	30687	0.141	0.920
3	4295	4096	30869	0.139	0.912
4	4380	4293	30907	0.142	0.918
5	4330	4347	30395	0.142	0.917
6	3664	3561	31960	0.115	0.847

(b)

Figure 6.3: Experiment results: seed customer: Base model A

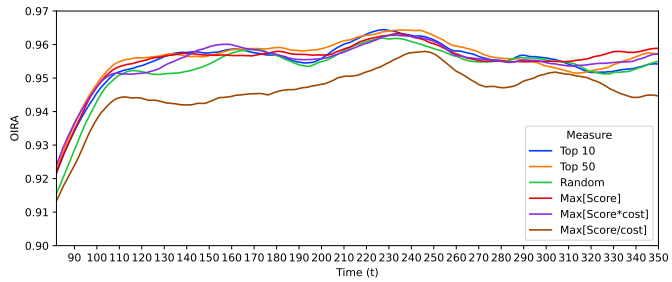


(a)

Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
7	3455	4238	30064	0.115	0.933
8	3411	4359	29965	0.114	0.932
9	3409	4059	30132	0.113	0.927
10	3428	4456	30204	0.113	0.933
11	3485	4232	29530	0.118	0.930
12	2751	3231	30869	0.089	0.888

(b)

Figure 6.4: Experiment results: seed customer: Base model B

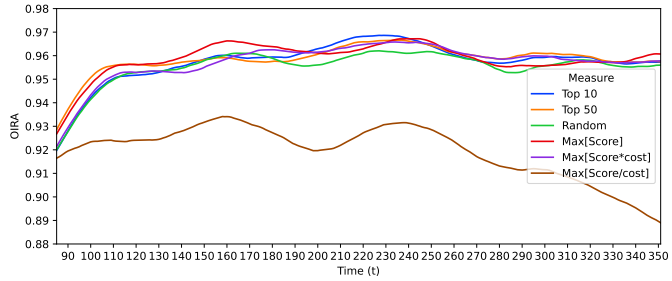


(a)

Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
13	2696	2714	33129	0.081	0.953
14	2681	2883	33033	0.081	0.954
15	2657	2732	33212	0.080	0.951
16	2712	2894	33248	0.082	0.954
17	2694	2892	32926	0.082	0.953
18	2644	2644	33471	0.079	0.944

(b)

Figure 6.5: Experiment results: seed customer: Base model C



(a)

Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
19	2485	2648	32347	0.077	0.955
20	2492	2794	32157	0.077	0.957
21	2465	2726	32214	0.077	0.953
22	2485	2793	32202	0.077	0.957
23	2492	2622	32070	0.078	0.955
24	2195	2198	32872	0.067	0.918

(b)

Figure 6.6: Experiment results: seed customer: Base model D

Looking at the OIRA graphs, we see that the performance of the experiments without seed customer rule (Experiments 6,12,18 and 24), perform considerably worse in comparison to the experiments with special seed customer rules. By looking at the KPIs total amount of inaccuracies found and the average OIRA for the experiments with special seed customer rules, we see that the random seed customer rule

performs the worst on each of the four base models regarding both selected KPIs. For the four other seed customer rules (Top 10, Top 50, Max[Score], and Max[Score*cost]) it is less clear which rule performs the best compared among each other.

From the OIRA graphs and the performance indicator table we can not see the effect of the four seed customer rules on the selection of locations in the warehouse. In Table 6.15, we provide the total times counted and the average failure length merged per five consecutive bays for base model A for these four seed customer rules. The average failure length are the average of the time length of failures that remain after the last counting day. By analysing Table 6.15 we found that the last five bays were the least time visited when we use seed customer rule 1 (exp 4) and most of the time visited when using seed customer rule 2 (exp 5). This also reflects the average failure length for the last five bays.

For the remaining experiments we select seed customer rule 5, because during the the third and fourth type of experiments we use other methods for calculating a score value. We want to make sure that locations further down the aisle are also selected during experiments with the other score calculation methods.

Bay	Exp 1	Exp 2	Exp 4	Exp 5
1-5	9471	8588	9506	8980
6-10	8092	7910	8260	7542
11-15	6976	7450	7185	7071
16-20	6193	6846	6064	6908
All	30732	30794	31015	30501

(a)

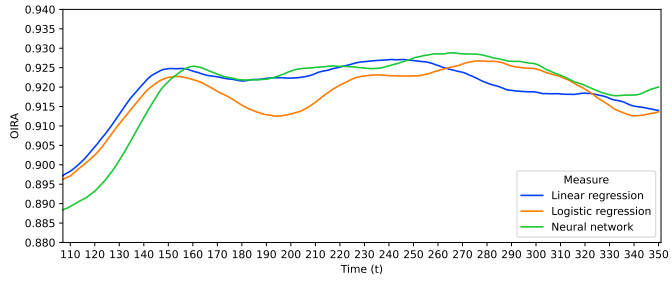
Bay	Exp 1	Exp 2	Exp 4	Exp 5
1-5	16.2	19.2	14.8	17.7
6-10	18.7	18.4	18.5	19.5
11-15	17.5	18.7	20.7	20.6
16-20	22.0	22.9	23.5	21.1
All	18.6	19.8	19.6	19.7

(b)

Table 6.15: Experiment results: seed customer: Base model A: total times counted per set of bays (a) and average length of failure per set of bays (b)

6.3.2 Benchmark learning models

We used seed customer rule 5 to conduct the second type of experiments. During the second type of experiments we compared the performance of using three different learning models for making the predictions E[P(inaccuracy)]. In the proposed model these predictions are used as score value (for the score-cost ratio), and for predicting the OIRA (POIRA). In Figure 6.7 and 6.8 we present the results of the experiments. The performance indicators are calculated over the period after the warm-up period, up to and including counting day 365.



(a)

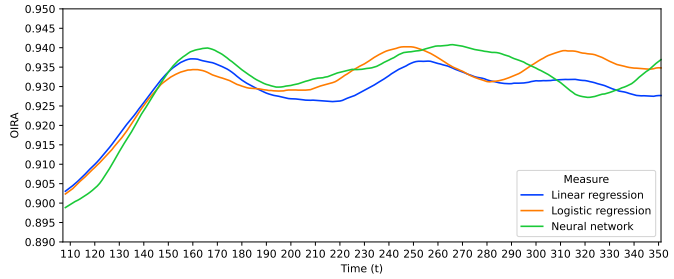
Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
25	4345	4381	30860	0.141	0.917
26	4333	4190	31065	0.139	0.916
27	4330	4347	30395	0.142	0.917

Exp.	$\overline{POIRA}$	MAD	$\overline{RE}$	$\overline{RMSE}$
25	0.920	0.027	0.347	0.050
26	0.906	0.018	0.288	0.043
27	0.920	0.011	0.365	0.034

(b)

Figure 6.7: Experiment results: benchmarking learning models: Base model A

First, we discuss the results in terms of how well it can find inaccurate inventory records. Thereafter, we discuss the quality of the predictions. For Base model A, using linear regression gives the best performance by looking at the total found and the $\overline{OIRA}$. However, in the OIRA graph we see that the NN line is above the linear regression and logistic regression line during the last 100 days. The NN has more trainable parameters in comparison to the linear and logistic regression models. We expect that due to



(a)

Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
28	3362	3348	29566	0.114	0.927
29	3443	3435	29883	0.115	0.930
30	3485	4232	29530	0.118	0.930

Exp.	$\overline{POIRA}$	MAD	$\overline{RE}$	$\overline{RMSE}$
28	0.943	0.028	0.553	0.111
29	0.925	0.019	0.497	0.104
30	0.926	0.014	0.958	0.101

(b)

Figure 6.8: Experiment results: benchmarking learning models: Base model B

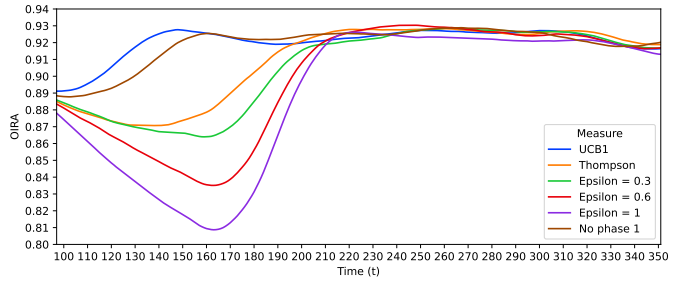
the the amount of parameters that the NN requires more training to provide good predictions, but with enough training that it can give better predictions. For Base model B, we found that the NN found in total the most inaccuracies and maintained the highest average OIRA (equal to logistic regression). However, looking at the OIRA graph, we see that the NN is not as dominant during the last 100 counting days as is the case for base model A.

Now we discuss the prediction performance of the three learning models. By looking for both Base model A as Base model B, we see that the $\overline{POIRA}$ of each of the learning models is relatively close to the $\overline{OIRA}$ of the experiments. By looking at the MAD, the average of the individual absolute deviation between OIRA and POIRA, we see that for both models the NN performs the best for predicting the accuracy of the warehouse. The relative error represents the relative deviation between amount found and amount expected to find during each cycle count. We see that for the NN that for both models the error is the highest. We further looked into the data and found that the NN tends to overpredict the amount of inaccuracies it is going to find. This is supported by the fact that the heuristic model wants to maximise the sum of the expected amounts. When we look at the average RMSE we see that the NN performs better in comparison to the other two methods.

In other words, the NN performs the best on predicting the P(inaccuracy) of all warehouse locations, but it performs the worst on predicting the amount it is going to find per cycle count moment. We expect that the amount of outliers in terms of prediction error are higher for logistic regression and linear regression in comparison to the NN, but that the height of the outliers are higher for the NN.

6.3.3 Introduction of phase I

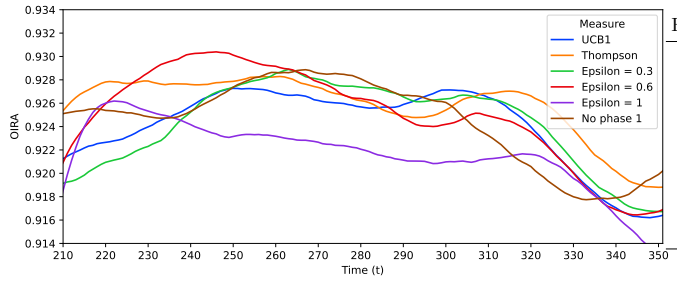
During the third type of experiments, we looked into the effect of implementing various methods for phase I (learning phase). We set the length of phase I to 90 days. In this section we present the results over the whole period (phase I and phase II) and we present the results for the period 210 to 350 for base model A and the period 220 to 350 for base model B. We selected these periods, since the decline in OIRA due to phase I, for each of the methods has disappeared as can be seen in Figure 6.9 and Figure 6.10. Note that Found(II), Predicted(II), and $\overline{OIRA}$(II) are calculated from the start of phase II which is cycle counting day 170 for base model A and day 180 for base model B.



(a)

Exp.	Found	Found(II)	Pred.(II)	$\overline{OIRA}$(I)	$\overline{OIRA}$(II)
31	4330	2866	2832	0.910	0.923
32	4282	3090	2921	0.878	0.923
33	4285	3212	2862	0.874	0.919
34	4218	3386	2976	0.861	0.916
35	4249	3575	3280	0.844	0.910
36	4330	2888	2946	0.904	0.924

(b)

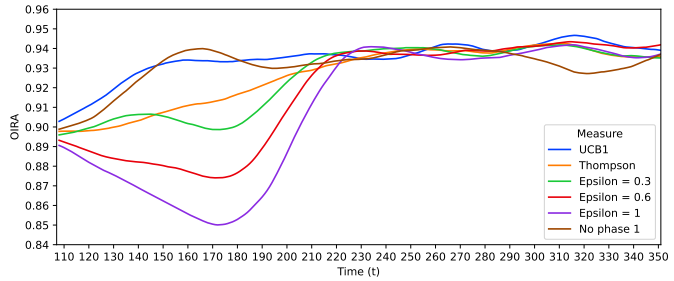


(c)

Exp.	$\overline{OIRA}$	$\overline{POIRA}$	MAD	Found	Pred.	$\overline{RE}$	$\overline{RMSE}$
31	0.924	0.926	0.0059	2058	2039	0.226	0.028
32	0.926	0.925	0.0048	2074	2033	0.212	0.027
33	0.924	0.928	0.0054	2076	1960	0.241	0.027
34	0.925	0.928	0.0055	2066	1893	0.227	0.026
35	0.922	0.923	0.0057	2048	1967	0.241	0.029
36	0.924	0.925	0.0059	2060	2117	0.295	0.028

(d)

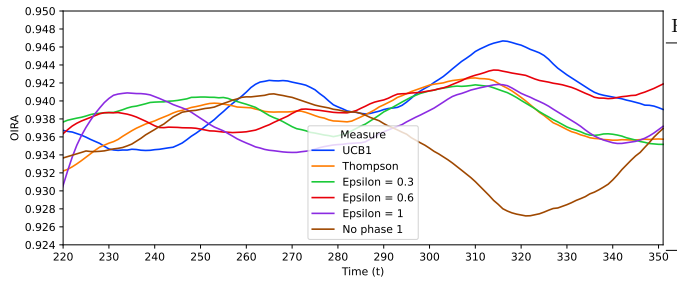
Figure 6.9: Experiment results: introduction of phase I: Base model A



(a)

Exp.	Found	Found(II)	Pred.(II)	$\overline{OIRA}$(I)	$\overline{OIRA}$(II)
37	3393	2276	3156	0.919	0.939
38	3334	2395	2784	0.903	0.935
39	3403	2532	3123	0.901	0.933
40	3348	2692	2898	0.885	0.930
41	3258	2735	2783	0.874	0.924
42	3499	2365	3011	0.918	0.935

(b)



(c)

Exp.	$\overline{OIRA}$	$\overline{POIRA}$	MAD	Found	Pred.	$\overline{RE}$	$\overline{RMSE}$
37	0.940	0.922	0.0194	1640	2330	1.509	0.094
38	0.938	0.924	0.0139	1709	1996	0.617	0.104
39	0.939	0.926	0.0122	1709	2247	0.764	0.089
40	0.939	0.936	0.0078	1717	2011	0.602	0.087
41	0.937	0.863	0.0745	1753	1923	0.481	0.091
42	0.935	0.926	0.0112	1690	2165	1.067	0.093

(d)

Figure 6.10: Experiment results: introduction of phase I: Base model B

In Figure 6.9a, and Figure 6.10a we see a drop in OIRA for four out of the five score methods in comparison to not using a score method during phase I. Only for the UCB-1 method the development of the OIRA during phase I is comparable to not using a score method. Also the average OIRA during phase II is comparable to each other. In Figure 6.9c, and Figure 6.10c, we see no clear best trained model for either base model A or base model B.

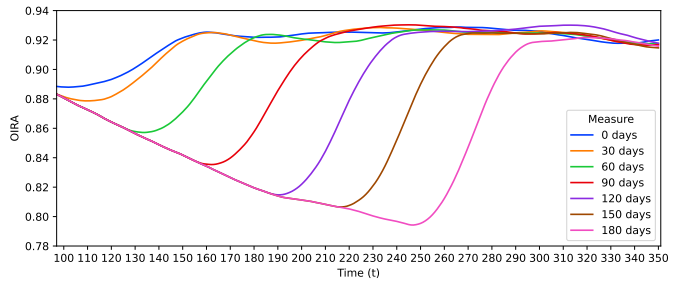
The goal of using a phase I is to increase the quality of the predictions. For base model A we see lower error rates MAD, $\overline{RE}$, and $\overline{RMSE}$ in comparison to base model B. For base model A, experiment 32 tends to perform the best by comparing the error rates. For base model B the fluctuation in error rates are higher in comparison to base model A. By looking at the error scores MAD and $\overline{RMSE}$ we see

that experiment 40 tend to be the best trained model. Since the effect of learning is more clear for base model B, we choose for the next set of experiments to use Epsilon = 0.6 for calculating the score value during phase I.

6.3.4 Length of phase I

During the fourth set of experiments we compared the effect of the length of phase I on the quality of the predictions. In Figure 6.11 and Figure 6.12 we present the results of the various experiments. For both base model A and base model B we analysed the performance over the entire cycle count (plot (a) and table (b)) and the performance over the last 100 days (plot (c) and table (d)).

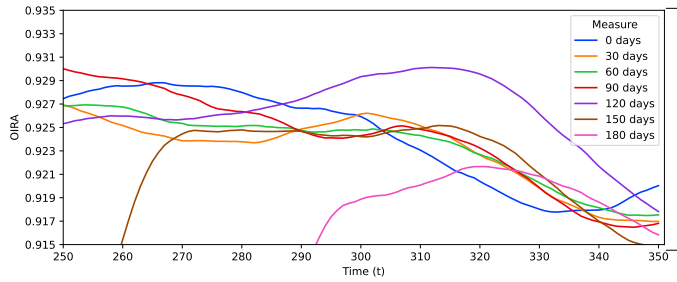
The OIRA graphs over the whole cycle counting length show that the decline in OIRA performance continues when the length of phase I increases. In both Table 6.11d as Table 6.12d, we see the lowest error rates and the highest average OIRA performance for the last 100 days. Therefore we choose to set the length of phase I to 120 days during the sensitivity analysis.



(a)

Exp.	Found	Found(II)	Pred.	Pred.(II)	Count	$\overline{OIRA}$
43	4330	4330	4347	4347	30395	0.917
44	4372	4051	4054	3732	30511	0.915
45	4297	3705	3969	3422	30457	0.908
46	4218	3386	3748	2976	30335	0.899
47	4180	3062	3734	2717	30345	0.887
48	4128	2660	3598	2300	30335	0.874
49	4063	2310	3654	2098	30245	0.860

(b)

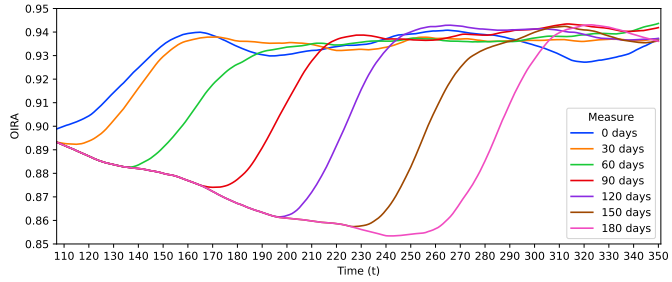


(c)

Exp.	$\overline{OIRA}$	$\overline{POIRA}$	MAD	Found	Pred.	$\overline{RE}$	$\overline{RMSE}$
43	0.924	0.925	0.0061	1487	1504	0.283	0.028
44	0.923	0.926	0.0057	1486	1448	0.241	0.026
45	0.923	0.926	0.005	1463	1495	0.257	0.024
46	0.924	0.926	0.0053	1472	1389	0.238	0.026
47	0.926	0.930	0.0049	1479	1403	0.171	0.025
48	0.922	0.929	0.0077	1595	1418	0.207	0.027
49	0.891	0.907	0.0167	2183	1983	0.229	0.046

(d)

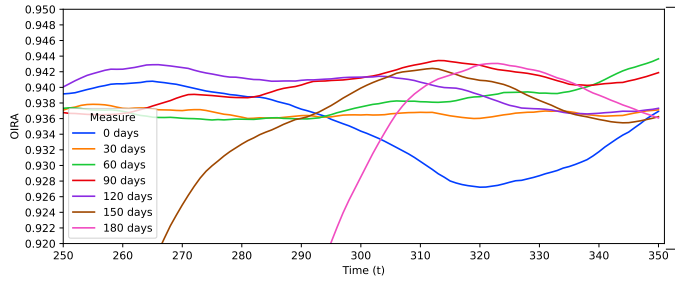
Figure 6.11: Experiment results: length phase I: Base model A



(a)

Exp.	Found	Found(II)	Pred.	Pred.(II)	Count	$\overline{OIRA}$
50	3499	3499	4232	4232	29637	0.930
51	3431	3147	4244	3925	29633	0.929
52	3427	2870	3730	3175	29495	0.924
53	3348	2538	3571	2791	29322	0.917
54	3212	2166	3202	2211	29215	0.909
55	3223	1948	3185	1970	29205	0.898
56	3059	1509	3041	1564	29069	0.890

(b)



(c)

Exp.	$\overline{OIRA}$	$\overline{POIRA}$	MAD	Found	Pred.	$\overline{RE}$	$\overline{RMSE}$
50	0.935	0.923	0.0137	1195	1626	1.162	0.092
51	0.937	0.921	0.0161	1239	1700	1.052	0.093
52	0.938	0.933	0.0087	1223	1423	0.71	0.090
53	0.940	0.934	0.009	1235	1465	0.622	0.087
54	0.940	0.940	0.0043	1169	1251	0.539	0.086
55	0.932	0.934	0.0064	1514	1611	0.684	0.098
56	0.911	0.921	0.0143	1579	1599	0.647	0.136

(d)

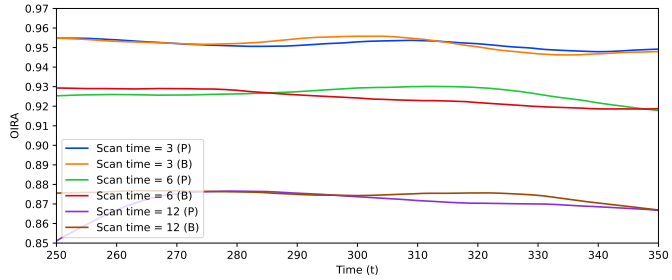
Figure 6.12: Experiment results: length phase I: Base model B

6.3.5 Sensitivity analysis

In this section we present the results of the sensitivity analysis. In each experiment we used the seed customer rule and the phase I scoring method, which we selected based on the previous set of experiments. For each of the experiments we analyse the performance of the model for the last 100 counting days, expect for the last set of experiments where we compare the OIRA estimates of the entire cycle counting period. For each of the experiments we set the length of phase I to 120 days except for the last set of experiments where we do not use a phase I.

Scanning time and flying time factor

In Figure 6.13, we show the results of our proposed method in comparison to a benchmark policy for various settings of the scan time. Here it can be seen that the experiment results of the benchmark policy are comparable to the results of our proposed method, given the same experimental settings.



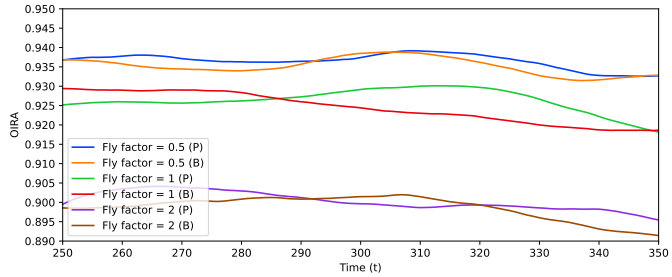
(a)

Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
57	1541	1531	17913	0.086	0.951
58	1533	N/A	16588	0.092	0.952
59	1479	1403	10833	0.137	0.926
60	1468	N/A	10344	0.142	0.924
61	1513	1217	5921	0.256	0.872
62	1381	N/A	5892	0.234	0.875

(b)

Figure 6.13: Experiment results: sensitivity analysis: scan time

In Figure 6.14, we show the results of our proposed method in comparison to a benchmark policy for various settings of the flying time factor time. Here it can be seen that the experiment results of the benchmark policy are comparable to the results of our proposed method, given the same experimental settings.



(a)

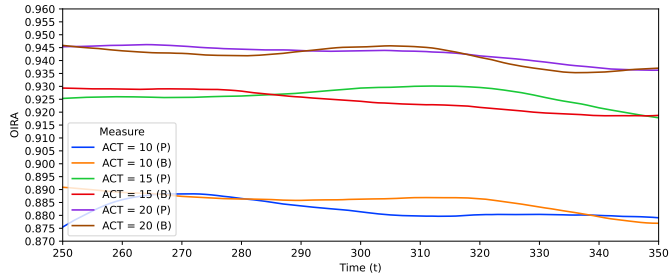
Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
63	1513	1521	12666	0.119	0.937
64	1552	N/A	12375	0.125	0.935
65	1479	1403	10833	0.137	0.926
66	1468	N/A	10344	0.142	0.924
67	1471	1339	8312	0.177	0.901
68	1494	N/A	7518	0.199	0.899

(b)

Figure 6.14: Experiment results: sensitivity analysis: flying time factor

Available counting time

In Figure 6.15, we show the results of our proposed method in comparison to a benchmark policy for various settings of the available counting time. Here it can be seen that the experiment results of the benchmark policy are comparable to the results of our proposed method, given the same experimental settings.



(a)

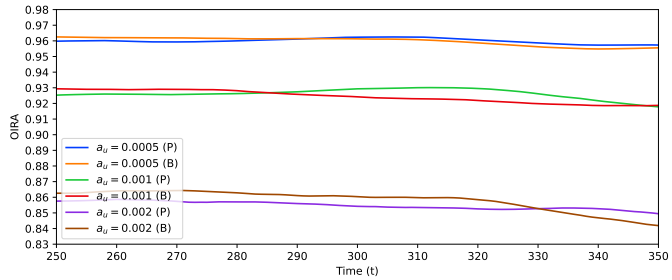
Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
69	1457	1203	7013	0.208	0.883
70	1372	N/A	6515	0.211	0.885
71	1479	1403	10833	0.137	0.926
72	1468	N/A	10344	0.142	0.924
73	1515	1520	14584	0.104	0.943
74	1541	N/A	14139	0.109	0.941

(b)

Figure 6.15: Experiment results: sensitivity analysis: available counting time

Failure rate weights

In Figure 6.16, we show the results of our proposed method in comparison to a benchmark policy for various settings of the failure rate weights. Here it can be seen that the experiment results of the benchmark policy are comparable to the results of our proposed method, given the same experimental settings.



(a)

Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
75	747	801	10792	0.069	0.960
76	753	N/A	10326	0.073	0.960
77	1479	1403	10833	0.137	0.926
78	1468	N/A	10344	0.142	0.924
79	2708	2497	10896	0.249	0.855
80	2729	N/A	10338	0.264	0.858

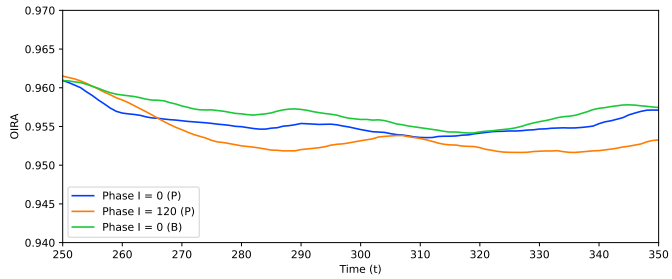
(b)

Figure 6.16: Experiment results: sensitivity analysis: failure rate weights

Layout

In Figure 6.17 and Figure 6.18, we show the results of our proposed method in comparison to a benchmark policy for various settings of the warehouse layout. Here it can be seen that the experiment results of the

benchmark policy are comparable to the results of our proposed method, given the same experimental settings.

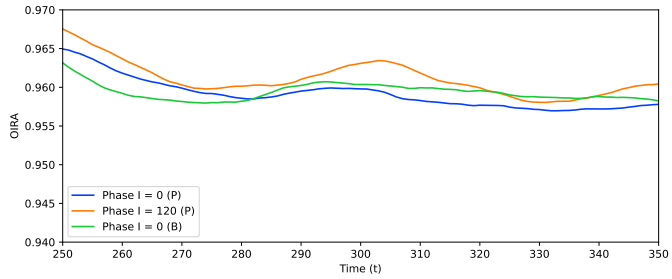


(a)

Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
81	904	897	11124	0.081	0.955
82	899	834	11165	0.081	0.954
83	898	N/A	10774	0.083	0.957

(b)

Figure 6.17: Experiment results: sensitivity analysis: warehouse layout (Base model C)



(a)

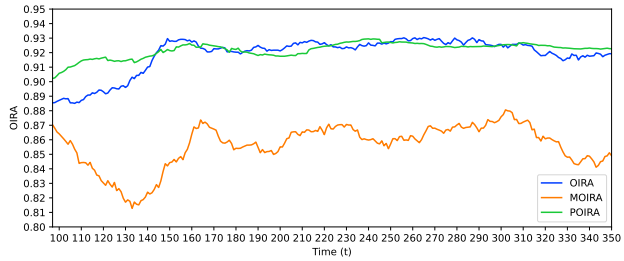
Exp.	Found	Predicted	Count	Found/count	$\overline{OIRA}$
84	792	801	10941	0.072	0.959
85	776	822	10897	0.071	0.961
86	800	N/A	10388	0.077	0.959

(b)

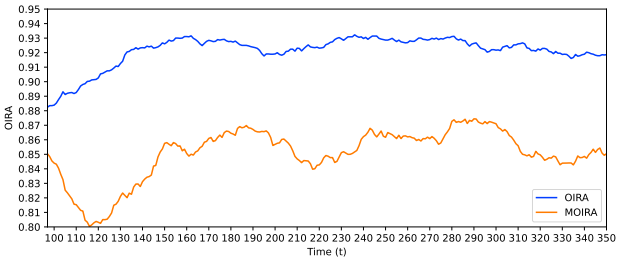
Figure 6.18: Experiment results: sensitivity analysis: warehouse layout (Base model D)

OIRA estimations

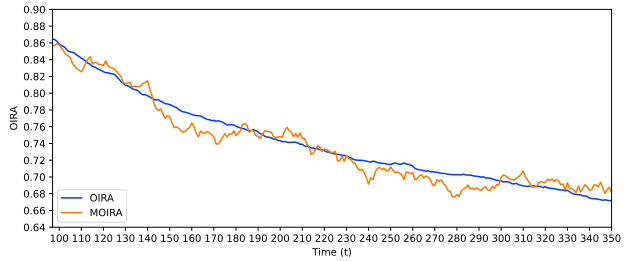
In Figure 6.19 and Figure 6.20, we show the results of our proposed method in comparison to the progress-based (B) and random benchmark policy (R) in terms of how well the models can forecast the actual OIRA. For both benchmark policies we calculated the monthly moving average of the MOIRA and compared this to the actual OIRA. For our proposed method we calculated the monthly moving average of both the MOIRA and the POIRA. For progress-based (B) and our proposed method we see in both Figure 6.19 and Figure 6.20, that the MOIRA does not perform well in terms of predicting the OIRA. For the random sampling method (R) we found that the MOIRA is much closer to the actual OIRA. Also the POIRA predictions of the proposed method are close to the actual OIRA. By looking at the MAD(MOIRA) of each of the experiments we see that the random sampling (R) performs much better in comparison to the other two methods. The MAD(POIRA) values of the proposed method are on the other hand much lower than those of the MOIRA. Next to that, MAD(POIRA) values are comparable to the MAD(OIRA) values of the random sampling benchmark (R).



(a)



(b)

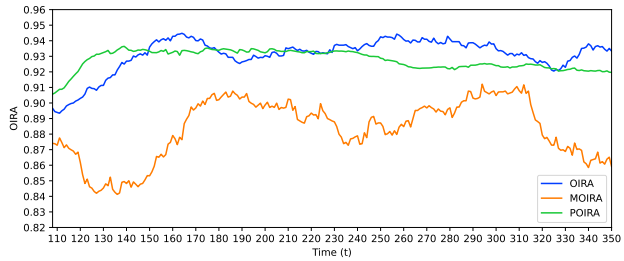


(c)

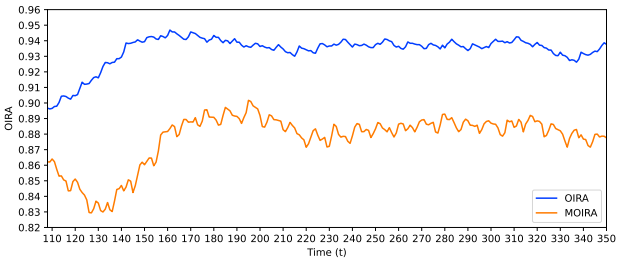
Exp.	Found	$\overline{OIRA}$	MAD(MOIRA)	MAD(POIRA)
87	4330	0.917	0.0627	0.00630
88	4325	0.920	0.0708	N/A
89	2049	0.745	0.00997	N/A

(d)

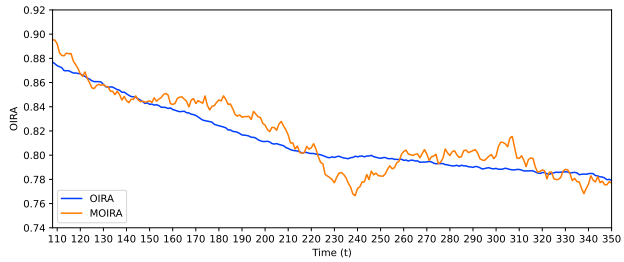
Figure 6.19: Experiment results: sensitivity analysis base model A: (a) proposed method, (b) benchmark B, (c) benchmark R, and (d) KPI results experiments



(a)



(b)



(c)

Exp.	Found	$\overline{OIRA}$	MAD(MOIRA)	MAD(POIRA)
90	3499	0.930	0.0478	0.00909
91	3463	0.932	0.0568	N/A
92	1410	0.815	0.00973	N/A

(d)

Figure 6.20: Experiment results: sensitivity analysis base model B: (a) proposed method, (b) benchmark B, (c) benchmark R, and (d) KPI results experiments

Chapter 7

Conclusion, discussion, and recommendations

7.1 Conclusion

This research is initialised by Bolk logistics and AIDA research group as part of a larger research for setting up an autonomous UAV solution for cycle counting. This research focused on the autonomous construction of cycle counting routes, where the performance of the cycle counts are taken into account. Therefore, we defined our main research question as:

"How to design a generic (operational) cycle counting method for a single UAV with time restricted counting intervals?"

The performance of cycle counts can be expressed in terms of the ability of finding inaccurate inventory records (1), or the ability to give good estimates of the warehouse accuracy (2). In our research we defined the performance in terms of the ability to find inaccurate records, but we took the latter into account.

We conducted a literature review where we found insights into often used existing cycle counting policies. Here we found that the performance of a cycle count depends on the trade-off between the amount of locations visited per cycle count moment and the likelihood that the inventory record status is inaccurate per location. Next to that, we found that some of these policies required knowledge about the failure behaviour of the warehouse to be able to get a good cycle counting performance, while at the introduction of an UAV to a new warehouse, the UAV has no understanding of this failure behaviour. To be able to deploy an autonomous UAV to a warehouse, a learning model is required, such that it gets this understanding.

A learning model can be implemented into the cycle counting process by using historic cycle counting data to train and test the learning model. This data can consist of multiple input parameters such as inputs about the storage locations, the stored goods and the transactions. The more types of input parameters the richer the data. Next to the input parameters, a learning model requires output labels to train and test. So, the historic cycle counting data should be labelled whether or not an inventory record status was *accurate*. To make predictions about the chance that the status of a location is *inaccurate*, the learning model requires the same type of input parameter data, such that it can predict the chances based on the current status of the warehouse.

We proposed a cycle counting method with the objective to maximise the sum of the predicted chance of being inaccurate of locations visited each cycle count. The proposed method is a 5 step greedy heuristic, that selects locations based on the highest score-cost ratio (score = E[P(inaccuracy)], cost = additional travel time + scantime). We made a simulated warehouse based on the warehouse layout and historic transaction data of Bolk logistics.

We tested our proposed cycle counting method in the simulated warehouse. Here we conducted experi-

ments with various seed customer rules (1), multiple learning models (2), various methods for calculating the *score* value of the score-cost ratio during the learning phase (3), the length of the learning phase (4), and we conducted a sensitivity analysis (5), where we altered various simulation model settings. During the first four type of experiments, we compared the results of various model settings among each other, while during the fifth type of experiments we compared the results of two benchmark cycle counting policies found in Chapter 2.

From these experiments, we concluded the following:

1. The selected seed location highly influences the direction of the constructed cycle counting route and which location neighbourhoods are visited. Especially, for locations further from the DS an appropriate seed location rule is required such that the locations are visited.
2. Using a neural network for predicting the chance that a location's inventory record is inaccurate, tends to give better predictions on the long-term in comparison to linear and logistic regression.
3. For all (except UCB-1) exploration-exploitation methods the OIRA drop during the learning phase, but the prediction errors after the learning phase are lower in comparison to not using a learning phase. For the UCB-1 method the OIRA increases during the learning phase but the prediction errors after the learning phase are not lower in comparison to not using a learning phase.
4. Using a special learning phase can help to increase the prediction performance of the neural network on the long-term. However, the cost of a gain of OIRA performance is high and the gain decreases over time.
5. (a) The proposed cycle counting method gives a comparable result in terms of OIRA performance to a benchmark location-based cycle counting policy for each of the sensitivity analysis experiments, but it significantly outperforms a pure random sampling policy.
   (b) The proposed cycle counting method significantly outperforms the location-based benchmark policy in terms of estimating the OIRA of the warehouse. It even tends to slightly outperform the random sampling policy in terms of estimating the OIRA.

7.2 Discussion and further research recommendations

Unfortunately, our more complex proposed model is not able to outperform the simple progress-based benchmark policy in terms of our objective to find as many as possible inaccuracies as defined in this research. We expect that the two most important reasons why the benchmark gives comparable results are: the simulated failure rate (1), and the definition of our objective function (2). Below, we first provide some arguments for each of the two reasons. Thereafter, we provide some suggestions for further research.

For Base model A, C, and D we used the same failure rate settings (Setting 1). After analysing Base model A, we found that during each cycle count day around 95% of all locations have a failure rate between 0.22% and 0.45% (Appendix D.2). The chance that those locations are still accurate after 50 days is between 80 and 90 per cent (Appendix D.1). Where 50 days is the approximate required time for the location-based benchmark policy to count every location. This lack of spread makes it less beneficial to visit certain locations more often than other locations, while this is the fundamental idea behind our proposed model.

For Base model B we had different failure rate settings (Setting 2). In Appendix D.3 and D.4, the distribution of this failure rate is shown and a table is presented for the chance that a location status is *accurate* after not being count for n consecutive cycle counting periods given a certain failure rate λ. As is shown in Appendix D.3, there is a clear spread in low failure rate, medium failure rate and high failure rate locations. The size of the high failure rate group is around 5% of all locations and the medium group is around 35%. Here we found that our proposed model benefits from visiting certain locations more often than others. However, the performance does not seem to be better compared to our benchmark. We analysed some experiments of Base model B (without a learning period). We focused on the remaining inaccuracies after the last count moment. The last moment since these locations where count is on average 45 days, where more than 40% of the remaining failures are not count during the last 50 counting days and more than 15% is not count during the last 70 days, while more than 95% of these location have a high or medium failure rate.

Our objective is to visit as much as possible inaccurate locations. We assume that an inaccurate location stays inaccurate until the location is visited. In progress-based cycle counting we visit each location once and after we visited each location, we start again. Eventually, we always find the inaccurate records, no matter how long the period is since the failure occurred.

Based on the previous paragraphs and some other research limitations and findings, we propose four main research topics for future research. These topics are the objective function (1), the failure rate function (2), construction of the routes (3) and score models for the learning phase (4). Below, per research topic we discuss some recommendations for further research:

Objective function

- We propose to redefine the objective function such that the length of an inaccuracy is penalised.
- We propose to allocate a cost factor to the inaccuracies which takes the severity of an inaccuracy into account. This makes the model more applicable in practice.

Failure function

- Testing the proposed cycle counting method in practice can overcome the limitation of our simulated failure rate function. Our failure rate function is simulated as a linear sum of individual parameter values, which are not correlated. We expect that in practice the failure function is more complex and can contain correlations. This will make it harder for the Neural network to learn the failure behaviour of the warehouse and this can increase the necessity of a learning phase. This also overcomes our problem of creating a representative failure rate.

Route construction

- We proposed five seed customer rules such that locations further down the aisles are selected. The seed customer initialises the direction in which a cycle counting route is constructed. During the construction of the cycle counting route we want to maximise our objective by adding locations with the highest score-cost ratio. If the seed customer directs to a neighbourhood with low predictions, then our model can construct a route where our objective is far from optimised. We propose to research the effect of constructing multiple routes per cycle counting moment, where the route with the highest objective value is executed.
- We propose to research the effect of a minimum and maximum for the interval of two consecutive counts of a location.

Learning phase

- During the learning phase, each cycle count moment we assign a score to each location using an exploration-exploitation technique. By doing this, we assume that a location's failure rate depends only on the warehouse location. However, the failure rate of the warehouse depends on multiple parameter values, where the contribution of a certain parameter value to the failure rate is simulated constant. The location's parameter values change due to transactions in the warehouse and therefore the failure rate of the location also changes. Therefore, we propose to construct an exploration-exploitation model that predicts scores based on parameter values instead of the warehouse location, such that it is possible to reduce the model's uncertainty about the effect of the parameter values instead of calculating an uncertainty score based on its location in the warehouse.

Bibliography

[1] Badri Adhikari. *[AI] Logistic regression using tensorflow keras*. 2020. URL: https://www.youtube.com/watch?v=KEYgPOcqmsw (visited on 03/04/2022).

[2] Opex Analytics. *Vehicle Routing Problems 101*. 2019. URL: https://medium.com/opex-analytics/opex-101-vehicle-routing-problems-262a173f4214 (visited on 02/06/2022).

[3] S.P. Anbuudayasankar, K. Ganesh, and K. Mohandas. *Mixed-Integer Linear Programming for Vehicle Routing Problem with Simultaneous Delivery and PickUp with Maximum Route-Length*. 2008. URL: https://scholarworks.waldenu.edu/cgi/viewcontent.cgi?article=1020&context=ijamt (visited on 02/21/2022).

[4] APICS. *APICS Extra Live: Constructing a Probability-Based Cycle Counting System*. 2021. URL: https://www.youtube.com/watch?v=5wFxRL3UVbQ (visited on 04/06/2022).

[5] Omar Besbes, Yonatan Gur, and Assaf Zeevi. *Optimal Exploration-Exploitation in a Multi-armed-Bandit Problem with Non-stationary Rewards*. Tech. rep. Stanford University, Graduate School of Business, 2014.

[6] Bhavesh Bhatt. *simple-linear-regression-tensorflow2.0*. 2020. URL: https://github.com/bhattbhavesh91/simple-linear-regression-tensorflow2.0/blob/master/regression-tf-notebook.ipynb (visited on 03/04/2022).

[7] Alex Birkett. *When to Run Bandit Tests Instead of A/B/n Tests*. 2019. URL: https://cxl.com/blog/bandit-tests/ (visited on 03/07/2022).

[8] Bolk transport B.V. Nederland. *Geschiedenis*. n.d. URL: https://bolk.nl/over-bolk/algemene-informatie/geschiedenis/ (visited on 03/19/2021).

[9] Harika Bonthu. *An Introduction to Logistic Regression*. 2021. URL: https://www.analyticsvidhya.com/blog/2021/07/an-introduction-to-logistic-regression/ (visited on 03/03/2022).

[10] Eleonora Bottani et al. "Inventory management in the presence of inventory inaccuracies: an economic analysis by discrete-event simulation". In: *International Journal of Supply Chain and Inventory Management* 2.9 (2017), pp. 39–73.

[11] Roger B. Brooks and Larry W. Wilson. "The Three-Phase Approach to Inventory Record Accuracy". In: *Inventory Record Accuracy*. John Wiley & Sons, Ltd, 2012. Chap. 3, pp. 29–33.

[12] Jason Brownlee. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. 2017. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (visited on 03/10/2022).

[13] Jason Brownlee. *How to Choose an Activation Function for Deep Learning*. 2021. URL: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (visited on 03/10/2022).

[14] Jason Brownlee. *Use Early Stopping to Halt the Training of Neural Networks At the Right Time*. 2018. URL: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/ (visited on 03/10/2022).

[15] Bradley James Bryant. *How to Calculate the Probability of Combinations*. 2017. URL: https://sciencing.com/calculate-number-combinations-5142125.html (visited on 02/24/2022).

[16] Flytbase Inc.. *White paper - Drone Automation for Warehouse 4.0*. Tech. rep. Flytbase, 2019.

[17] Michael J. Garbade. *Regression Versus Classification Machine Learning: What's the Difference?* 2018. URL: https://medium.com/quick-code/regression-versus-classification-machine-learning-whats-the-difference-345c56dd15f7 (visited on 03/03/2022).

[18] Catherine Gitau. *Classification in Supervised Machine Learning: All you need to know!* 2018. URL: https://categitau.medium.com/in-one-of-my-previous-posts-i-introduced-machine-learning-and-talked-about-the-two-most-common-c1ac6e18df16 (visited on 03/03/2021).

[19] John D. Hedengren. *Knapsack Optimization*. 2021. URL: https://apmonitor.com/me575/index.php/Main/KnapsackOptimization (visited on 03/03/2022).

[20] Anne Marie Helmenstine. *Absolute and Relative Error and How to Calculate Them*. 2021. URL: https://sciencenotes.org/absolute-and-relative-error-and-how-to-calculate-them/ (visited on 03/10/2022).

[21] HBM Prenscia Inc. *Derivations of Failure Rate Equations for Series and Parallel Systems*. 2016. URL: https://www.weibull.com/hotwire/issue181/article181.htm (visited on 03/07/2022).

[22] Quanterion Solutions Incorporated. *Mission Reliability and Logistics Reliability: A Design Paradox*. 2021. URL: https://www.quanterion.com/mission-reliability-and-logistics-reliability-a-design-paradox/ (visited on 03/08/2022).

[23] Johan Joubert and SJ Claasen. "A sequential insertion heuristic for the initial solution to a constrained vehicle routing problem". In: *ORiON* 22 (2006).

[24] Tackseung Jun. "Survey on the bandit problem with switching costs". In: *De Economist* 152 (2004), pp. 513–541.

[25] Yun Kang and Stanley B. Gershwin. "Information inaccuracy in inventory systems: stock loss and stockout". In: *IIE Transactions* 37.9 (2005), pp. 843–859.

[26] Ajitesh Kumar. *Micro-average & Macro-average Scoring Metrics – Python*. 2020. URL: https://vitalflux.com/micro-average-macro-average-scoring-metrics-multi-class-classification-python/ (visited on 03/08/2022).

[27] Richard C. Larson, Amedeo R. Odoni, and Arnold Barnett. *Lecture notes - Some Important Heuristics for the TSP*. Massachusetts Institute of Technology. 2006.

[28] Gabriele De Luca. *SVM Vs Neural Network*. 2021. URL: https://www.baeldung.com/cs/svm-vs-neural-network (visited on 03/03/2022).

[29] Shie Mannor. "k-Armed Bandit". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 561–563.

[30] Guilherme Duarte Marmerola. *Introduction to Thompson Sampling: the Bernoulli bandit*. 2017. URL: https://gdmarmerola.github.io/ts-for-bernoulli-bandit/ (visited on 03/07/2022).

[31] Tim Massey. *A guide to the basics of successful material handling*. 2017. URL: https://www.flexqube.com/news/guide-basics-successful-material-handling/ (visited on 10/08/2021).

[32] James D. McCaffrey. *Why a Neural Network is Always Better than Logistic Regression*. 2017. URL: https://jamesmccaffrey.wordpress.com/2018/07/07/why-a-neural-network-is-always-better-than-logistic-regression/ (visited on 03/03/2022).

[33] Jaroslav Menčík. *Reliability of Systems*. 2016. URL: https://www.intechopen.com/chapters/50094 (visited on 03/08/2022).

[34] Martijn R.K. Mes, Johannes M.J. Schutten, and Arturo Eduardo Perez Rivera. "Inventory routing for dynamic waste collection". In: *Waste management* 34.9 (2014), pp. 1564–1576.

[35] Cory Mitchell. *How To Use a Moving Average to Buy Stocks*. 2022. URL: https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp (visited on 03/10/2022).

[36] P. Molema. *Inventory count*. 2018. URL: https://www.toolshero.com/financial-management/inventory-count/ (visited on 03/10/2022).

[37] Derrick Mwiti. *Keras Loss Functions: Everything You Need to Know*. 2021. URL: https://neptune.ai/blog/keras-loss-functions (visited on 03/10/2022).

[38] Simon P. Neill and M. Reza Hashemi. "Chapter 8 - Ocean Modelling for Resource Characterization". In: *Fundamentals of Ocean Renewable Energy*. Academic Press, 2018, pp. 193–235.

[39] Micro Focus International plc. *SVM (Support Vector Machine) for Regression*. 2021. URL: https://www.vertica.com/docs/9.2.x/HTML/Content/Authoring/AnalyzingData/MachineLearning/SVM/SVM__SupportVectorMachine_forRegression.htm (visited on 03/04/2022).

[40] Warren B. Powell. "Clearing the Jungle of Stochastic Optimization". In: *Bridging Data and Decisions*. Informs, 2014. Chap. 4, pp. 109–137.

[41] Warren B. Powell. *Sequential decision analytics and modeling: modeling exercises with python*. 2021. URL: https://tinyurl.com/sequentialdecisionanalytics (visited on 06/19/2021).

[42] Warren B. Powell and Ilya O. Ryzhov. *Optimal learning 2nd edition (Draft)*. John Wiley & Sons inc., 2018.

[43] M. Rossetti, T. Collins, and Ravi Kurgund. "Inventory Cycle Counting – A Review". In: *The proceedings of the 2001 Industrial Engineering Research Conference*. Vol. 1. 2001, pp. 457–463.

[44] Pranshu Sharma. *Training Neural Network with Keras and basics of Deep Learning*. 2021. URL: https://www.analyticsvidhya.com/blog/2021/11/training-neural-network-with-keras-and-basics-of-deep-learning/ (visited on 03/10/2022).

[45] Luis Da Silva. *Reinforcement learning basics: stationary and non-stationary multi-armed bandit problem*. 2019. URL: https://towardsdatascience.com/reinforcement-learning-basics-stationary-and-non-stationary-multi-armed-bandit-problem-cfe06d33b815 (visited on 03/21/2021).

[46] Simplilearn. *Neural Network In 5 Minutes — What Is A Neural Network? — How Neural Networks Work — Simplilearn*. 2019. URL: https://www.youtube.com/watch?v=bfmFfD2RIcg (visited on 03/06/2022).

[47] Simul8. *Warm Up Period*. n.d. URL: https://www.simul8.com/support/help/doku.php?id=features:clock:warm_up_period (visited on 09/16/2021).

[48] Ian Smith. *Case study - Inventory automation*. 2020. URL: https://www.ware.ai/blog/case-study-warehouse-automation-roi (visited on 05/25/2021).

[49] Valamis. *Learning Curve Theory*. 2021. URL: https://www.valamis.com/hub/learning-curve (visited on 11/29/2021).

[50] Thom Veneman. "Scheduling algorithm for autonomous robot cycle counting". MA thesis. Rijksuniversiteit Groningen, 2019.

[51] Verity AG. *Warehouse-automation*. 2021. URL: https://verity.ch/warehouse-automation/ (visited on 03/11/2021).

[52] Luc Wijffels et al. "An enhanced cycle counting approach utilising historical inventory data". In: *IFAC-PapersOnLine* 49.12 (2016), pp. 1347–1352.

[53] Elaine Zhang. *Comparing Multi-Armed Bandit Algorithms on Marketing Use Cases*. 2018. URL: https://towardsdatascience.com/comparing-multi-armed-bandit-algorithms-on-marketing-use-cases-8de62a851831 (visited on 03/07/2022).

Appendices

Appendix A

Initial balance correction examples

Time	Type	Balance
0	Initial balance	1
14	Out	0
14	In	1
32	Out	0
35	In	1
64	Out	0
65	In	1
70	Out	0
78	Out	-1
81	In	0
106	Out	-1
107	In	0
115	Out	-1
115	In	0
130	Out	-1
132	In	0
169	Out	-1
174	In	0
185	Out	-1
190	In	0
309	Out	-1
311	In	0
352	Out	-1
356	In	0
365	Out	-1

Table A.1: All ingoing and outgoing transactions with balance location: H 09001 03

Time	Type	Balance
0	Initial balance	1
16	Out	0
17	In	1
42	Out	0
44	In	1
127	Out	0
127	In	1
130	Out	0
236	Out	-1
236	In	0
246	Out	-1
249	In	0
275	Out	-1
279	In	0
282	Out	-1
282	In	0
283	Out	-1
316	Out	-2
316	In	-1
318	Out	-2
318	In	-1
325	Out	-2
326	In	-1
339	Out	-2
344	In	-1

Table A.2: All ingoing and outgoing transactions with balance location: D 01003 01

Time	Type	Balance
0	Initial balance	1
8	Out	0
14	In	1
24	Out	0
30	In	1
63	Out	0
64	In	1
86	Out	0
88	In	1
113	Out	0
115	In	1
129	Out	0
135	In	1
228	Out	0
228	In	1
235	Out	0
237	In	1
249	Out	0
251	In	1
260	Out	0
262	In	1
275	Out	0
279	In	1
281	Out	0
282	In	1
282	Out	0
284	Out	-1
285	In	0
290	Out	-1
297	Out	-2
299	In	-1
326	Out	-2
331	In	-1

Table A.3: All ingoing and outgoing transactions with balance location: D 01003 02

Appendix B

Distance model verification examples

B.1 Example 1

Manual calculations

Horizontal movement:

1. most left aisle = 1
2. Distance DS most left aisle = 7
3. Furthest pallet location = 54 (Bay 18, Location on shelf 3)
4. Distance moved in aisle = 108 (2 times furthest pallet)
5. Distance last aisle to DS = 7 (only one aisle visited)
6. total horizontal movement = $7 + 108 + 7 = 122$

Vertical movement:
No vertical movement.

Total movement:
There is no vertical movement. So the total movement is equal to the horizontal movement, which is 122.

Result Python model
We used the Python indices from Table B.1 as input for our sample distance calculations (Figure B.1). In Figure B.2 the result of the calculation is shown. The model calculated that the length of the sample route is 122, which is equal to our example calculation above.

Sample no.	Location code	Python index no.
1	K 02003 01	4841
2	L 12003 01	5561
3	K 14003 01	5129
4	L 18003 01	5705

Table B.1: Bolk location codes example 1

```
SampleArrayChecker[0]= 4841
SampleArrayChecker[1]=5129
SampleArrayChecker[2]=5561
SampleArrayChecker[3]=5705
while SampleDistance < SampleTime:
    SampleArray.append(SampleArrayChecker[i])
    SampleDistance= CalculateDistances(SampleArray[i],i+1)
    i = i +1
    if i == 4:
        print('The length of the sample route =',int(SampleDistance))
DistancePerRound.append(SampleDistance)
Samplesize= len(SampleArray)
AmountCountedPerRound.append(Samplesize)
```

Figure B.1: Example 1: input

```
IPdb [2]: !continue
The length of the sample route = 122
```

Figure B.2: Example 1: result

B.2 Example 2

Manual calculations

Horizontal movement:

1. most left aisle = 1
2. Distance DS most left aisle = 7
3. Furthest pallet location aisle 1 = 54 (Bay 18, Location on shelf 3)
4. Distance moved in aisle 1= 108 (2 times furthest pallet)
5. next aisle = 6
6. Distance to next aisle = 15 $((6-1)*3)$
7. Furthest pallet location aisle 6 = 36 (Bay 12, Location on shelf 3)
8. Distance moved in aisle 6 = 72 (2 times furthest pallet)
9. Distance last aisle to DS = 8
10. total horizontal movement = $7+108+15+72+8=210$

Vertical movement:
No vertical movement.

Total movement:
There is no vertical movement. So the total movement is equal to the horizontal movement, which is 210.

Result Python model
We used the Python indices from Table B.2 as input for our sample distance calculations (Figure B.3). In Figure B.4 the result of the calculation is shown. The model calculated that the length of the sample route is 210, which is equal to our example calculation above.

Sample no.	Location code	Python index no.
1	K 02003 01	4841
2	L 12003 01	5561
3	K 14003 01	5129
4	L 18003 01	5705
5	A 02003 01	41
6	B 03003 01	545
7	A 07003 01	161
8	B 12003 01	761

Table B.2: Bolk location codes example 2

```
SampleArrayChecker[0]= 4841
SampleArrayChecker[1]=5129
SampleArrayChecker[2]=5561
SampleArrayChecker[3]=5705
SampleArrayChecker[4]=41
SampleArrayChecker[5]=545
SampleArrayChecker[6]=161
SampleArrayChecker[7]=761
while SampleDistance < SampleTime:
    SampleArray.append(SampleArrayChecker[i])
    SampleDistance= CalculateDistances(SampleArray[i],i+1)
    i = i +1
    if i == 8:
        print('The length of the sample route =',int(SampleDistance))
DistancePerRound.append(SampleDistance)
Samplesize= len(SampleArray)
AmountCountedPerRound.append(Samplesize)
```

Figure B.3: Example 2: input

```
IPdb [2]: !continue
The length of the sample route = 210
```

Figure B.4: Example 2: result

B.3 Example 3

Manual calculations

Horizontal movement:

1. most left aisle = 1
2. Distance DS most left aisle = 7
3. Furthest pallet location aisle 1 = 54 (Bay 18, Location on shelf 3)
4. Distance moved in aisle 1= 108 (2 times furthest pallet)
5. next aisle = 2
6. Furthest pallet location aisle 2 = 24 (Bay 8, Location on shelf 3)
7. Distance moved in aisle 2 = 48 (2 times furthest pallet)
8. next aisle = 3
9. Furthest pallet location aisle 3 = 60 (Bay 20, Location on shelf 3)
10. Distance moved in aisle 3 = 120 (2 times furthest pallet)
11. next aisle = 4

12. Furthest pallet location aisle 4 = 51 (Bay 17, Location on shelf 3)
13. Distance moved in aisle 4 = 102 (2 times furthest pallet)
14. next aisle = 5
15. Furthest pallet location aisle 5 = 24 (Bay 8, Location on shelf 3)
16. Distance moved in aisle 5 = 48 (2 times furthest pallet)
17. next aisle = 6
18. Furthest pallet location aisle 6 = 36 (Bay 12, Location on shelf 3)
19. Distance moved in aisle 6 = 72 (2 times furthest pallet)
20. Distance to first aisle to last aisle = 15 $((6-1)*3)$
21. Distance last aisle to DS = 8
22. total horizontal movement = $7+108+48+120+102+48+72+15+8 = 528$

Vertical movement:
No vertical movement.

Total movement:
There is no vertical movement. So the total movement is equal to the horizontal movement, which is 528.

Result Python model
We used the Python indices from Table B.3 as input for our sample distance calculations (Figure B.5). In Figure B.6 the result of the calculation is shown. The model calculated that the length of the sample route is 528, which is equal to our example calculation above.

Sample no.	Location code	Python index no.
1	K 02003 01	4841
2	L 12003 01	5561
3	K 14003 01	5129
4	L 18003 01	5705
5	I 04003 01	3929
6	J 08003 01	4505
7	G 20003 01	3353
8	E 01003 01	1937
9	F 16003 01	2777
10	E 17003 01	2321
11	C 08003 01	1145
12	A 02003 01	41
13	B 03003 01	545
14	A 07003 01	161
15	B 12003 01	761

Table B.3: Bolk location codes example 3

```
SampleArrayChecker[0]= 4841
SampleArrayChecker[1]=5129
SampleArrayChecker[2]=5561
SampleArrayChecker[3]=5705
SampleArrayChecker[4]=3929
SampleArrayChecker[5]=4505
SampleArrayChecker[6]=3353
SampleArrayChecker[7]=1937
SampleArrayChecker[8]=2777
SampleArrayChecker[9]=2321
SampleArrayChecker[10]=1145
SampleArrayChecker[11]=41
SampleArrayChecker[12]=545
SampleArrayChecker[13]=161
SampleArrayChecker[14]=761


while SampleDistance < SampleTime:
    SampleArray.append(SampleArrayChecker[i])
    SampleDistance= CalculateDistances(SampleArray[i],i+1)
    i = i +1
    if i == 15:
        print('The length of the sample route =',int(SampleDistance))
DistancePerRound.append(SampleDistance)
Samplesize= len(SampleArray)
AmountCountedPerRound.append(Samplesize)
```

Figure B.5: Example 3: input

```
(IPdb [2]): !next
The length of the sample route = 528
```

Figure B.6: Example 3: result

B.4 Example 4

Manual calculations

Horizontal movement:

1. most left aisle = 1
2. Distance DS most left aisle = 7
3. Furthest pallet location = 54 (Bay 18, Location on shelf 3)
4. Distance moved in aisle = 108 (2 times furthest pallet)
5. Distance last aisle to DS = 7 (only one aisle visited)
6. total horizontal movement = $7 + 108 + 7 = 122$

Vertical movement:

1. Starting height = 1 (DS)
2. Height first location = 4 (Rack 2, Bay 2)
3. Height second location = 8 (Rack 1, Bay 12)
4. Height third location = 2 (Rack 2, Bay 14)
5. Height fourth location = 5 (Rack 1, Bay 18)
6. Difference DS and first location = 3
7. Difference first and second location = 4

8. Difference second and third location = 6
9. Difference third and fourth location = 3
10. Difference fourth (last) location and DS = 4
11. total vertical movement = $3 + 4 + 6 + 3 + 4 = 20$

Total movement:
The total movement is the sum of the vertical movement and the horizontal movement. So the total movement is $20 + 122 = 142$.

Result Python model
We used the Python indices from Table B.4 as input for our sample distance calculations (Figure B.7). In Figure B.8 the result of the calculation is shown. The model calculated that the length of the sample route is 142, which is equal to our example calculation above.

Sample no.	Location code	Python index no.
1	K 02003 04	4844
2	L 12003 08	5568
3	K 14003 02	5130
4	L 18003 05	5709

Table B.4: Bolk location codes example 4

```
SampleArrayChecker=random.sample(list(LocDf['LocationsNumber']),LocAmount)
SampleArrayChecker[0]= 4844
SampleArrayChecker[1]=5130
SampleArrayChecker[2]=5568
SampleArrayChecker[3]=5709



while SampleDistance < SampleTime:
    SampleArray.append(SampleArrayChecker[i])
    SampleDistance= CalculateDistances(SampleArray[i],i+1)
    i = i +1
    if i == 4:
        print('The length of the sample route =',int(SampleDistance))
DistancePerRound.append(SampleDistance)
Samplesize= len(SampleArray)
AmountCountedPerRound.append(Samplesize)
```

Figure B.7: Example 4: input

```
((((((((((((IPdb [2])))))))))))): !next
the length of the sample route =  142
```

Figure B.8: Example 4: result

Appendix C

Code for implementation of the neural network model using Keras library

```
def PredictInaccChance(TrainData_Df, AllLocInfo_Df):
    #Random shuffle the train dataframe
    np.random.shuffle(TrainData_Df.values)
    #parameters: 1 = article, 2 = operator, 3= shift number, 4 = Height, 5= Bay number 6 = balance
    TrainDataDf = TrainData_Df.sample(frac=0.8, random_state= 25)
    TestDataDf = TrainData_Df.drop(TrainDataDf.index)


    if NoInputsNeu == 6:
        #Make numpy formats x and y from the TrainData_Df
        TrainSet_x = np.column_stack((TrainDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[0,1]],
                                      TrainDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[1,1]],
                                      TrainDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[2,1]],
                                      TrainDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[3,1]],
                                      TrainDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[4,1]],
                                      TrainDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[5,1]],
                                      TrainDataDf.TimeSinceLastCount.values))
        TestSet_x = np.column_stack((TestDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[0,1]],
                                      TestDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[1,1]],
                                      TestDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[2,1]],
                                      TestDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[3,1]],
                                      TestDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[4,1]],
                                      TestDataDf.iloc[:,SelectedNeuralNetworkParameters.iat[5,1]],
                                      TestDataDf.TimeSinceLastCount.values))
        #Make predictionsarray
        PredictArray = np.column_stack((AllLocInfo_Df.iloc[:,SelectedNeuralNetworkParameters.iat[0,2]],
                                        AllLocInfo_Df.iloc[:,SelectedNeuralNetworkParameters.iat[1,2]],
                                        AllLocInfo_Df.iloc[:,SelectedNeuralNetworkParameters.iat[2,2]],
                                        AllLocInfo_Df.iloc[:,SelectedNeuralNetworkParameters.iat[3,2]],
                                        AllLocInfo_Df.iloc[:,SelectedNeuralNetworkParameters.iat[4,2]],
                                        AllLocInfo_Df.iloc[:,SelectedNeuralNetworkParameters.iat[5,2]],
                                        AllLocInfo_Df.Time_since_last_Count.values))

    #Make numpy formats x and y from the TrainData_Df
    TrainSet_y = TrainDataDf.Status.values
    TestSet_y = TestDataDf.Status.values
    monitor_val_loss = keras.callbacks.EarlyStopping(monitor='val_loss', patience = 50, restore_best_weights=True)

    if NeuralNetworkTensor == 1:
        #Make the neural network model
        model = keras.Sequential([
        keras.layers.Dense(2, input_shape=((NoInputsNeu+1),), activation='sigmoid'),
        keras.layers.Dense(32, activation='sigmoid'),
        keras.layers.Dense(2, activation='softmax')
        ])
        #compile the neural network model
        model.compile(optimizer= keras.optimizers.Adam(0.01),
                  loss=keras.losses.SparseCategoricalCrossentropy(from_logits=False),
                  metrics=['accuracy'])
        #Fit the model to the train data set
        model.fit(TrainSet_x ,TrainSet_y , batch_size=480, epochs= 500, verbose=0,
        callbacks=[monitor_val_loss], validation_data=(TestSet_x, TestSet_y))

    #Make predictions
    PredictionsAllLocs = np.round(model.predict(PredictArray),3)
    return PredictionsAllLocs
```

Appendix D

Failure rate distribution and chance of status *accurate* after n days

D.1 Chance of location status *accurate* after n days given certain failure rate λ

	λ											
Days	0.00225	0.00250	0.00275	0.00300	0.00325	0.00350	0.00375	0.00400	0.00425	0.00450	0.00475	0.00500
1	0.998	0.998	0.997	0.997	0.997	0.997	0.996	0.996	0.996	0.996	0.995	0.995
2	0.996	0.995	0.995	0.994	0.994	0.993	0.993	0.992	0.992	0.991	0.991	0.99
3	0.993	0.993	0.992	0.991	0.99	0.99	0.989	0.988	0.987	0.987	0.986	0.985
4	0.991	0.99	0.989	0.988	0.987	0.986	0.985	0.984	0.983	0.982	0.981	0.98
5	0.989	0.988	0.986	0.985	0.984	0.983	0.981	0.98	0.979	0.978	0.976	0.975
6	0.987	0.985	0.984	0.982	0.981	0.979	0.978	0.976	0.975	0.973	0.972	0.97
7	0.984	0.983	0.981	0.979	0.977	0.976	0.974	0.972	0.971	0.969	0.967	0.966
8	0.982	0.98	0.978	0.976	0.974	0.972	0.97	0.968	0.967	0.965	0.963	0.961
9	0.98	0.978	0.976	0.973	0.971	0.969	0.967	0.965	0.962	0.96	0.958	0.956
10	0.978	0.975	0.973	0.97	0.968	0.966	0.963	0.961	0.958	0.956	0.954	0.951
15	0.967	0.963	0.96	0.956	0.952	0.949	0.945	0.942	0.938	0.935	0.931	0.928
20	0.956	0.951	0.946	0.942	0.937	0.932	0.928	0.923	0.918	0.914	0.909	0.905
25	0.945	0.939	0.933	0.928	0.922	0.916	0.91	0.905	0.899	0.893	0.888	0.882
30	0.935	0.928	0.921	0.914	0.907	0.9	0.893	0.887	0.88	0.873	0.867	0.86
35	0.924	0.916	0.908	0.9	0.892	0.885	0.877	0.869	0.862	0.854	0.846	0.839
40	0.914	0.905	0.896	0.887	0.878	0.869	0.86	0.852	0.843	0.835	0.827	0.818
45	0.904	0.893	0.883	0.874	0.864	0.854	0.844	0.835	0.826	0.816	0.807	0.798
50	0.893	0.882	0.871	0.861	0.85	0.839	0.829	0.818	0.808	0.798	0.788	0.778

Table D.1: Chance of location status *accurate* after n days given certain failure rate λ

D.2 Distribution of λ for Base model A failure rate setting 1

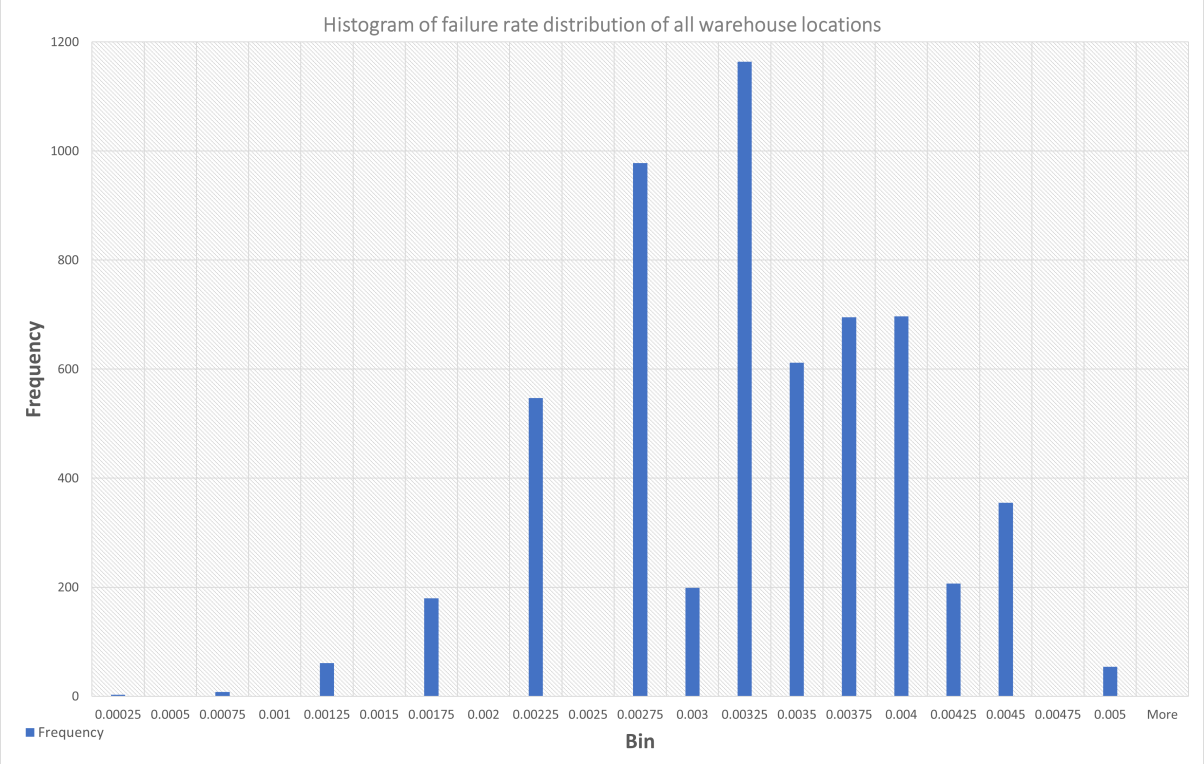


Figure D.1: Distribution of λ for Base model A failure rate setting 1

D.3 Distribution of λ for Base model B failure rate setting 2

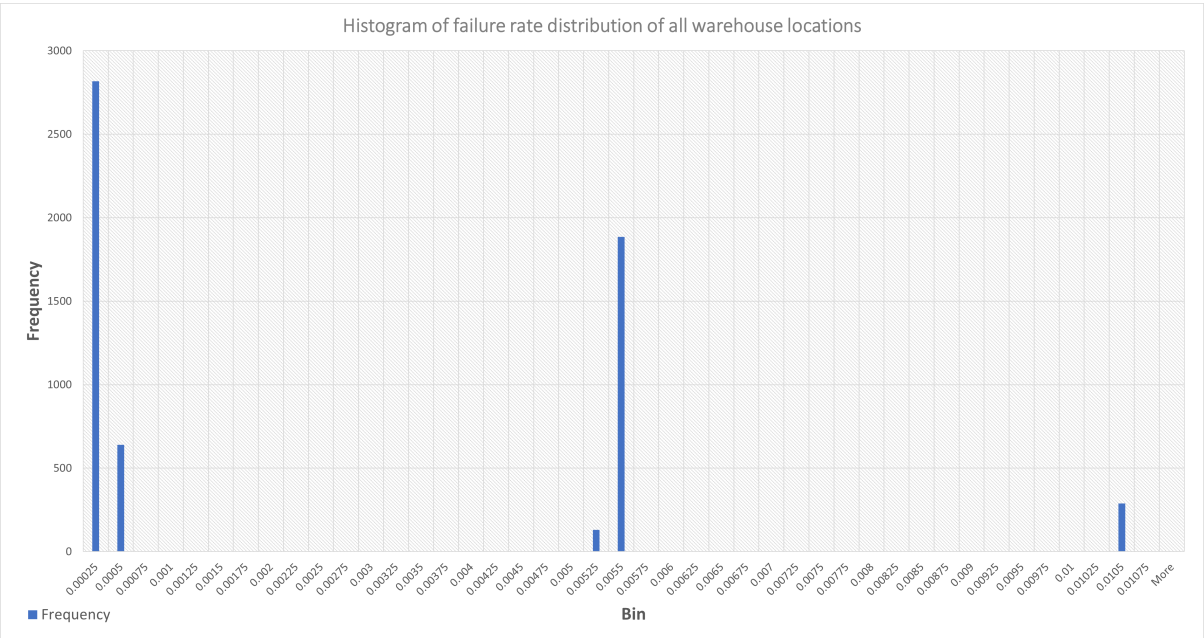


Figure D.2: Distribution of λ for Base model B failure rate setting 2

D.4 Chance of location status *accurate* after n days given certain failure rate λ

	λ				
Days	0.0002	0.0005	0.0052	0.0055	0.0105
1	0.9998	0.9995	0.9948	0.9945	0.9895
2	0.9995	0.9990	0.9895	0.989	0.9791
3	0.9993	0.9985	0.9843	0.9836	0.9688
4	0.9990	0.9980	0.9792	0.9782	0.9587
5	0.9988	0.9975	0.9740	0.9728	0.9486
6	0.9985	0.9970	0.9689	0.9675	0.9386
7	0.9983	0.9965	0.9638	0.9621	0.9288
8	0.9980	0.9960	0.9588	0.9568	0.9190
9	0.9978	0.9955	0.9537	0.9516	0.9094
10	0.9975	0.9950	0.9487	0.9463	0.8998
15	0.9963	0.9925	0.9241	0.9206	0.8536
20	0.9950	0.9900	0.9001	0.8956	0.8097
25	0.9938	0.9876	0.8767	0.8712	0.7681
30	0.9925	0.9851	0.8539	0.8475	0.7286
35	0.9913	0.9826	0.8317	0.8245	0.6911
40	0.9900	0.9802	0.8101	0.8020	0.6556
45	0.9888	0.9777	0.7891	0.7802	0.6219
50	0.9876	0.9753	0.7686	0.7590	0.5899

Table D.2: Chance of location status *accurate* after n days given certain failure rate λ