University Utrecht

Technical Artificial Intelligence

Master Thesis

Automatic Detection Of Suspicious Behaviour

Author:
Iris Renckens

Supervisors:
Dr. Selmar Smit
Dr. Ad Feelders
Prof. Dr. Arno Siebes

September 2, 2014

Abstract

This thesis presents an implementation of an anomaly detection method that can be used for surveillance applications. The method is based on the perspective that a flying vehicle should be able to derive the objects in an environment that show anomalous behaviour. The objective is to make an method that is able to examine a very large urban environment in real time. This is done with a clustering algorithm that takes into account the paths, speed and lifetime of objects. The clusters are learned with only coordinates of the object in an environment. These clusters form the basis of the distance calculation in the detection phase. This method is based on a method proposed by Piciarelli et al. [14] where a number of important improvements have been made. With our method we solved some essential problems of the method of Piciarelli, such as the time dependency of the model and the amount of data needed to create a good model. We also added environmental features to be taken into account. These improvements resulted in a faster and better working method for this particular purpose.

Contents

1 | Introduction

1.1 Project Background

The background of this project comes from the simulation research group within TNO. They are involved in research for the ministry of Defence concerning the exploration of possibilities of UaV's (Unmanned aerial Vehicles). The purpose of their research is to investigate what is possible with these highly technical machines. Since security and surveillance is always an important topic, they are interested in a method that can find strange behaviour in a certain urban area with these UaV's. This originated from a project they did with ships at sea. It appeared that surveillance employees could not see all that was happening in the videos.

An example of this situation at the sea environment is the following scenario. There were two types of ships at sea, large vessels and small fishing boats. The large vessels appeared to always fare at a high speed and in a straight line. The fishing boats appeared to fare slowly and in strange turns, then they stop for a while to fish and then they move again. If at a certain moment in time a small ship is sailing at a very (unusual) high speed in a straight line towards a large vessel, it is not 'normal' behaviour for a fishing boat and it is too small to be a vessel, it seems to be a pirate ship.

It would be beneficial to have a system that attends surveillance operators to suspicious behaviour. It might not always be the case that this object is indeed strange, but it helps the operator to find the abnormal situations in the videos.

This can be achieved in many ways. In this case an unsupervised learning method is chosen. This means that with only the normal data as learning input, the method should know if new behaviour is unusual (more about different data inputs can be found in section 1.2.2). With this choice we create a model that can be used in any situation, it has no prior knowledge about the environment which makes it widely applicable.

1.2 Anomaly Detection Introduction

In this section we will review what anomalies are and how we can capture them. It will be a brief explanation about the different anomaly types, algorithms and domain applications. Most of this information is from the survey paper of Chandola et al.[4].

1.2.1 Introduction to Anomaly Detection

The term anomaly detection is used for the problem of finding patterns in data that do not conform to expected behaviour, these patterns are interesting to the analyst. These patterns may indicate suspicious persons or other sorts of noticeable behaviour.

The most important challenges to anomaly detection are:

- Defining a region which describes 'normal' behaviour. It is not always clear what these regions should be.
- Anomalous observations appear like normal, it is difficult to find them in the data.
- Normal behaviour keeps evolving. What might be normal behaviour now will not be the same at another point in time, so it is hard to capture.
- Normal is different for every different application domain. In every domain normal means some other behaviour.
- The availability of labelled data, this is necessary for training your model but not always available.
- Data contains noise which makes it harder to extract the anomalous behaviour since you don't want noise to be labelled as anomalous behaviour.

1.2.2 Different Aspects

Nature of input

The input of an anomaly algorithm is a collection of data instances. Every data instance is a set of attributes. These attributes can be different types such as binary, categorical or continuous. Every data instance can consist of only one attribute (*univariate*) or multiple attributes (*multivariate*).

The data instances can be related to each other in multiple ways:

- *Sequence data*: the data instances are linearly ordered (for example time series).
- *Spatial data*: each data instance is related to its neighbour (for example ecological data).
- *Graph data*: data instances are represented as a graph and are connected through vertices.

Type of anomaly

Anomalies can be classified in three categories, Point Anomalies, Contextual Anomalies and Collective Anomalies. They will be explained briefly.

- *Point Anomalies*: an individual data instance is anomalous with respect to the rest of the data
- *Contextual Anomalies*: a data instance is anomalous in a specific context, also known as conditional anomaly. Each data instance is defined according to these two attributes:
  - Contextual attributes: these determine the context or neighbourhood.
  - Behavioural attributes: these define the non-contextual characteristics.

  The anomalous behaviour is determined using the values for the behavioural attributes within a specific context. Another challenging factor is the availability of contextual attributes.
- *Collective Anomalies*: a collection of data instances are anomalous in relation to the complete data set. Collective anomalies can only occur in data sets where the data instances are related.

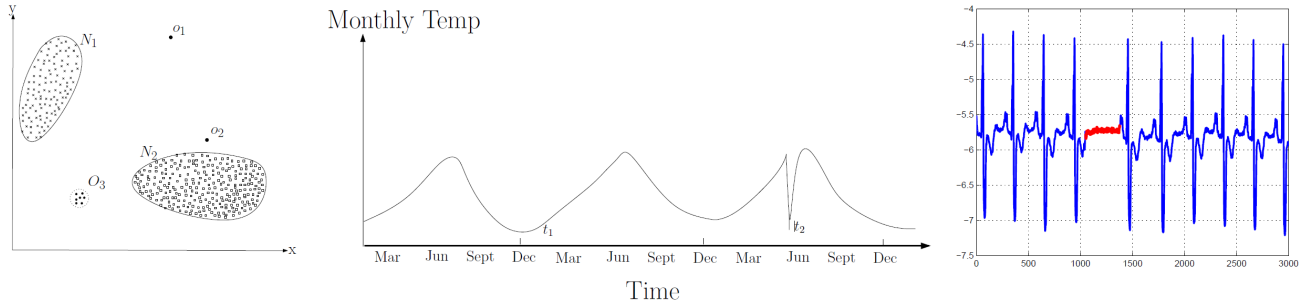


Figure 1.1: From left to right: Point Anomalies (o_1, o_2, O_3), Contextual Anomalies (t_2) and Collective Anomalies, taken from the paper of Chandola et al.[4]

Data labels

You want to know for every data instance to what class it belongs, whether it is 'normal' or anomalous. This labeling is often manually done by experts and is quite expensive. Based on the availability of these labels there are three modes to operate in:

- *Supervised anomaly detection*: this mode assumes the availability of training data with given labels for the normal class as well as for the anomalous class. A typical approach is to build a model that predicts the class label. There are two major issues in supervised anomaly detection: the anomalous instances are far fewer compared to the normal instances in the training data and obtaining accurate and representative labels, especially for the anomaly class is usually challenging.
- *Semi-supervised anomaly detection*: this mode assumes the training data has labelled instances for the normal class only.
- *Unsupervised anomaly detection*: this mode does not require any labelled training data. It makes the implicit assumption that normal instances are far more frequent than anomalies in the test data.

1.2.3 Applications

There are many application domains for anomaly detection namely:

- *Intrusion detection*: detection of malicious activity. The key challenge is the huge volume of data and the issue of a high false alarm rate due to such a large sized input.

  Applications:

  - Host Based Intrusion Detection Systems.
  - Network Intrusion Detection Systems.

- *Fraud detection*: detection of criminal activities occurring in commercial organizations. The challenge is that it requires on-line detection because it needs to be signalled as soon as it takes place.

  Applications:

  - Credit Card Fraud Detection.
  - Mobile Phone Fraud Detection.
  - Insurance Claim Fraud Detection.
  - Insider Trading Detection.

- *Medical and Public Health Anomaly Detection*: detecting abnormal patient condition, instrumentation errors, recording errors or disease outbreaks in a specific area. The challenge in this area is that the cost of classifying an anomaly as normal can be very high.
- *Industrial Damage Detection*: detecting damage due to continuous usage.

  Applications:

  - Fault Detection in Mechanical Units.
  - Structural Defect Detection.
- *Image Processing*: detecting changes over time or regions which appear abnormal on the static image. One of the main challenges is the size of the input.
- *Anomaly Detection in Text Data*: detects novel topics, events or news stories in a collection of documents or news articles.
- *Sensor Networks*: detects sensor faults or intrusions or both.
- Many more...

1.3 Methodology

The main research problem for this thesis is "How (and is) it possible to detect suspicious behaviour autonomously in an urban environment." This sections describes the phases and sub problems of the project.

In the first stage of the project we will look into how we can model behaviour. What are the parameters that are necessary to describe the behaviour so the system is able to learn it. We will start with an overview of the relevant literature. This literature study will be based on the field of anomaly detection and the characterisation of behaviour. We will first consider individual behaviour and eventually we might look into expanding the model for suspicious group behaviour. There are certain situations where the behaviour of an individual is not anomalous, but when a whole group behaves this way it is anomalous. We might also look into the environmental objects, some situations can be suspicious even without the presence of humans. An example can be strangely parked cars or a location where all the lights are suddenly turned off. *In this first section the sub problem is to map the different kind of methods for anomaly detection.*

The stage following is to determine the use of anomaly detection in UAV's. We will look at the data that can be captured and how this can be used in a anomaly detection method. We will also look at the different aspects that we can extract from the data and how the general form of the method should look. *In this second section the sub problem is to determine the use of anomaly detection for our particular setting.*

In the next stage we will look for a suitable method to solve our problem. There are many available anomaly detection methods, we will find the most suitable method and adjust it to our problem. We will take into account the different parameters that we found for modelling the behaviour and put it all together in a model. *In this third section the sub problem is to determine the most suitable anomaly detection method and adjust it for our purpose.*

This model will be implemented for testing purposes and it will be integrated in the TNO testing systems. This might take some effort because it should work under those specific circumstances. When everything is working in the embedded test setting we will do some test sessions. Afterwards an evaluation will be done about the system. We will look into the effectiveness of the method and what can be improved. *In this fourth section the sub problem is to evaluate the used method.*

1.4 Thesis Outline

This thesis is organised as follows:

Firstly we will give an overview on the current research in the field of anomaly detection in chapter 2. We look into the different kinds of anomaly detection techniques that exist nowadays and summarise the advantages and disadvantages of these techniques. In chapter 3 we will describe the sort of data that will be used as well as the characteristics for the operation system. The different features of the input are determined and the way the model will operate. Following this we will outline the exact details of the used method in the 4^{th} chapter. Firstly the method choice is explained to show which different kinds of methods could be used for this application and that clustering is the most suitable. Secondly we go into detail on clustering of single values and how this achieves the goals of the application. Then we go slightly in to detail on the background of track clustering which gives us a more informed view of the overall method used. Then we shall discuss the clustering method for the tracks where we will touch on the method proposed by Piciarelli et al. [14] as well as some alterations we have made to this proposal to make it work. The exact implementation of a few important functions is also discussed as well as different testing methods. In chapter 5 we discuss the specific additions we have made to the clustering method to ensure a better application. We also highlight several problems that we encountered during the development of these additions and the solutions that we have found. The topics are the environmental attributes that we added, the dependence of the model, the level of data needed for a good model and lastly we discuss the difficulty of the parameter settings that can be altered. Chapter 6 shows the simple test case to demonstrate the effectiveness of the application. Subsequently we show a more complex test case in which we use a traffic simulator to generate data that has a high resemblance to reality. With this test we are able to compare the improved method to the original method. Lastly we discuss the implications of our work in chapter 7 as well as future research that can be done to further improve the anomaly detection in large urban environments.

2 | Related Work

2.1 Related Literature

There are many papers that deal with anomaly detection [4]. In this thesis we focus on the anomaly detection in surveillance videos and all that belongs to this subject. It specifically includes anomaly detection of completely unknown environments, unlabelled data and clustering methods. For learning behaviour it includes capturing trajectories, motion patterns and the learning of a behavioural model.

2.1.1 Unlabelled Data

Most of the anomaly detection algorithms are based on the assumption that there are labelled instances for learning the model. This means that there is a fixed idea of what is called anomalous. In this project we deal with a fully unknown environment, so we will not know anything about the environment but also nothing about what can be seen as anomalous. This means that it will be quite a challenge to learn a good model. It is possible to take into account the environment when learning the normal behaviour.

A particular paper focuses on conditional anomaly detection [18], this method takes into account the environmental attributes to determine outliers. It uses a statistical model to decrease the amount of false positive rate during the on-line anomaly detection process. This is done with a Gaussian Mixture Model to determine the maximum likelihood of each state. Afterwards conditional anomaly detection is executed as follows: assume that each data point is a set of two attributes, an environmental attribute and an indicator attribute and use this environmental attribute as a grouping attribute. This way you can find outliers that appear to be normal without these extra attributes.

2.1.2 Sequence patterns

Other papers try to capture the anomalies in the sequence patterns only. This can be useful for anomaly detection in the walking patterns of the object. There are many ways to apply anomaly detection to sequences, an overview of this is given in [5]. Sequence patterns are used in [20] where anomalies are found in patterns of user behaviour in a system. This is done with Time-based Inductive Machine, it discovers temporal patterns from observations of a temporal process. These patterns are used to model user profiles. When checking for anomalies, the deviation from these profiles is determined.

Another specific method for motion behaviour is used in [8] where patterns of motions are modelled. It assumes different locations in an environment such as the bedroom, the bathroom and the kitchen, which it can apply the method to. The behaviour of humans is modelled as a sequence of these locations which will be registered by sensors. The learnt rules with a high enough support will provide a model for the behaviour. The challenge is to find the interesting sequences during post processing. This is done with three filters. The first is the time between two occurrences, closer in time makes the assumption that it belongs more together. The second filter removes redundant patterns by deleting all sub sequences with the same support. The third assumes that some sensors keep on registering the same activities.

2.1.3 Surveillance videos

A lot of papers are available for finding anomalies in surveillance videos. Some papers also build a model of 'normal' behaviour first and then determine the outliers with this model in real-time applications, as used in the paper of Brax et al. [2]. The model is based on the movement of the objects in the environment and the states of objects such as the state of the course and speed. For a new observation it searches in the learnt model for a similar result. It will return the observed situation as anomalous if it can not find a match or if the match is below a certain threshold.

Xiang et al.[21] proposes a framework for behaviour profiling and anomaly detection without any labelling of the training set. Grouping of behaviour is done through unsupervised model selection and feature selection on the eigenvectors of an affinity matrix.

In [16] they make use of the eigenvectors and the calculation of affinity matrices for the detection of anomalies. This paper also has an interesting view on the data that can be captured from objects, it defines some features that are important for modelling behaviour. This is interesting for our project where we also want to captures those features. You need to save the correct data to learn a good model of behaviour. It includes some possible features that can be captured from surveillance videos. These different features are stored and taken into account when finding the anomalies in the data.

There are more papers about features that can be used for modelling behaviour, in [3] the term observables is used for this purpose. This paper assesses the situation based on features. These features range from *carries object*, *group collision* to *empty square*. More about features can be found in section 3.2. With these features a probabilistic model is build that is based on a Hidden Markov Model.

2.1.4 Hidden Markov Model

When the data is captured, the behaviour needs to be modelled. Most papers use Hidden Markov Models (HMM) [17] as a modelling technique. All of the coordinates can be seen as states in an environment. The states in each network are drawn using a Gaussian probability function. This way the states of the behaviour are clustered together. These states are then used in the HMM's to learn the motion patterns. The paths are given as input to the HMM. A Hidden Markov Model makes use of different states it can be in, as seen in figure 2.1 as x. These states are 'hidden' hence the name. Going from one state to another happens with some probability a_{ij} for the state i to state j. All of the states emit some observable y that can be seen as the output. The probability of a certain output in a certain state is denoted by b. During the anomaly detection, the HMM determines a certain probability of the likelihood of a new pattern given the learnt patterns.

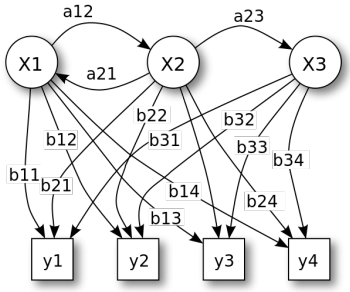


Figure 2.1: Hidden Markov Model (source: `wikipedia.org`)

This method has been used for example for learning movements in mobile networks [6]. Here the states were the different places in a network of mobile phone users. It determines the neighbours of each state and updates these relations by the amount of time a mobile phone user moved from a state to the other. These tracks are used to learn a HMM. This HMM was used for anomaly detection but also for state prediction.

It has also been used for learning behaviour patterns in every day life [12]. Here the HMM is also used to learn the normal trajectories of movements of humans. The HMM is then used during the anomaly detection to create a likelihood vector. This vector represents the probability of a behaviour given the learnt model.

2.1.5 Clustering

There are also examples of papers that use the learnt HMM to cluster the data [19]. Here a stream of data is used for on-line creating and learning of a HMM. For anomaly detection a new track provides a likelihood vector which indicates if it can be seen as normal. A clustering technique is used to cluster similar tracks together, this is done on-line, as the tracks expand. These tracks that are clustered together form the new set of states for the HMM, so the model is updated.

The same technique has been applied in [1] for time-series data. Here a challenge is to determine the amount of clusters. For this problem they use a method that tries to find a model with Minimum Description Length. The clustering itself is based on expectation maximisation (EM) and *k*-means.

In clustering it is important that you take an appropriate distance measure. This is different for each application. If you have a distance measure that is not suited to your data, it will not give you the desired results. A comparison between distance measures especially on trajectory clustering in surveillance videos in made in [22].

2.1.6 Trajectories

There are papers with proposals for anomaly detection especially for trajectories. A survey on this topic can be found in [13]. It discusses different techniques used for trajectory learning and their advantages and disadvantages.

One of the papers uses the patterns of motions for automatic learning [9]. The trajectories are hierarchically clustered using spatial and temporal information and then each motion pattern is represented as a chain of Gaussian distributions. With these distributions the anomalies can be determined. There is another paper that presents a statistically based model on object trajectories [10]. With these trajectories a model is learnt, which can be used for the identification of incidents, event recognition and trajectory prediction. Fu et al. [7] proposes a hierarchical clustering framework to classify motion trajectories. Then spectral clustering is used to group trajectories with similar spatial patterns. There is another paper that is based on single-class support vector machine clustering, where the novelty detection SVM capabilities are used for the identification of anomalous trajectories [15].

A different approach for the clustering of trajectories in video surveillance systems is done in [14]. Here the clustering is done in an on-line fashion as the data is collected. From these data points the distance is measured to the nearest cluster and if it deviates too much, a new cluster is formed. The clusters are linked together in the form of a tree where the edges are probabilities of following a certain path. It can be used for anomaly detection since you can calculate the likelihood of a certain path given the learnt 'normal' paths.

2.2 Anomaly Detection Techniques

In this section we will review most of the existing anomaly detection methods. Again, most of this is taken from the survey paper of Chandola et al.[4].

2.2.1 Classification Based Detection Techniques

Classification is used to learn a model (a classifier) from a data set. The model is first created from a set of labelled data instances, this is the training set, and then used to classify a test instance into one of the learnt classes, this is the testing phase.

An important assumption that is made: *A classifier that can distinguish between normal and anomalous classes can be learnt in the given feature space.*

Multi-class classification means that there are multiple normal classes, if an instance is not part of one of these, then it is considered an anomaly. One-class classification means there is a normal class and all the data instances that are not a part of this class are labelled anomalous.

Neural Network Based

Neural network based classification can be applied to multi-class and one-class classification.

For a multi-class classification a technique is used that trains a neural network on the normal training data to learn the different normal classes. Then each test instance is provided as an input for this neural network. If the network accepts this input it is normal, otherwise it is labelled as anomalous.

For the one-class classification another technique is used. *Replicator Neural Networks* is used to train the model into three hidden layers. The testing phase involves reconstruction of each data instance, then the calculation of the reconstruction error. This error is the anomaly score for this data instance.

Bayesian Network Based

Bayesian network classification is done in multi-class classification. A basic technique estimates the posterior probability of observing a class label using a naive Bayes network. The label with the highest probability will be chosen as class label.

This basic technique assumes independence between the different attributes, a more complex Bayesian network can be used to capture the correct conditional independences which can lead to better results.

Support Vector Machines Based

Support Vector Machines have been applied to anomaly detection in the one-class setting. Such techniques learn the region that is considered 'normal' with the training data. For every test instance is determined whether it is in this region. If the data instance is captured in this region it is considered normal, otherwise as anomalous.

Rule Based

Rule based anomaly detection techniques learn rules that capture the normal behaviour of a system. If there is no such rule for an instance, it is considered as anomaly. It can be applied to multi-class and one-class classification.

For multi-class classification there are two steps. The first step is to learn rules from the training data where every rule has an according confidence value. The second step is to find the rule that best captures the data instance. The inverse of the confidence value is the anomaly score.

For one-class classification association rules are used to generate rules in an unsupervised manner. To ensure that the rules are solid, a support threshold is used to prune rules with low support.

Advantages and disadvantages

+ It makes use of powerful algorithms that can distinguish between classes
+ Testing can be done fast because of the precomputed model
- It relies on the availability of accurate labels, which is not always possible
- In classification only, it assigns only a class, not a meaningful score

2.2.2 Nearest Neighbour Based Detection Techniques

An important assumption that is made with nearest neighbour based techniques is: *Normal data instances occur in dense neighbourhoods, while anomalies occur far from their closest neighbours.* This means if a data instance is far away for other data instances for a certain distance measure, it is likely to be anomalous.

Nearest neighbour based anomaly detection techniques require a distance or similarity measure defined between two data instances. For categorical attributes is often a matching coefficient used.

Nearest neighbour based techniques can be grouped into two categories:

- Techniques that use the distance of a data instance to its k^{th} nearest neighbour as the anomaly score.
- Techniques that compute the relative density of each data instance to determine its anomaly score.

Distance to K^{th} Nearest Neighbour

A basic nearest neighbour technique is based on the following definition: The anomaly score of a data instance is defined as its distance to its k^{th} nearest neighbour in a given data set.

There are some variants of this method:

- Modify the above definition to calculate the anomaly score as an amount of data instances.

  Another way to compute the anomaly score is to count the number of nearest neighbours in a certain radius. This can be seen as estimating the global density for every data instance.

- Use different distance/similarity measures to handle other data types.

  There are variant techniques proposed which uses techniques such as a hyper-graph based technique and a distance measure for data containing a mix of categorical and continuous attributes.

- Focus on improving the efficiency of the basic technique.

  Variant techniques that improve the efficiency are those that prune the search space by either ignoring instances that cannot be anomalous or by focussing on instances that are most likely to be anomalous.

Relative Density

Density techniques estimate the density of the neighbourhood of each data instance. If an instance is in a sparse neighbourhood it is declared anomalous and if it is in a dense neighbourhood it is declared normal.

Density based techniques perform poorly if the data has regions of varying densities. To handle the issue of varying densities in the data set, a set of techniques have been proposed to compute density of instances relative to the density of their neighbours.

A method for this is the Local Outlier Factor (LOF). For any given data instance, the LOF score is equal to the ratio of average local density of the k nearest neighbours of the instance and the local density of the data instance itself. There is a variant for this method called the Connectivity-based Outlier Factor (COF). In COF, the neighbourhood for an instance is computed in an incremental way. And there are even more of these variant methods.

Advantages and disadvantages

+ They are unsupervised in nature, they are purely data driven which makes it easy to use these data sets
+ Semi-supervised techniques perform better than unsupervised techniques in terms of missed anomalies
+ Adapting nearest neighbour based techniques to a different data type is straightforward, it is mostly about defining a appropriate distance measure
- For unsupervised techniques, if the data has normal instances that do not have enough close neighbours or if the data has anomalies that have enough close neighbours, the technique fails to label them correctly
- For semi-supervised techniques, if the normal instances in test data do not have enough similar normal instances in the training data, the false positive rate for such techniques is high

- The computational complexity of the testing phase is a significant challenge

- Performance of a nearest neighbour based technique greatly relies on a distance measure, it can be challenging to find a good measure when the data is complex

2.2.3 Clustering Based Detection Techniques

Clustering is used to group similar data instance together in clusters. Even though clustering appears to be very different from anomaly detection, there are several techniques developed with clustering. There are three categories of clustering based techniques.

The first category is based on the assumption: *Normal data instances belong to a cluster in the data, while anomalies do not belong to any cluster.* Techniques based on this apply a known clustering algorithm and classify all data instance that do not belong to one of the clusters as anomalous. A disadvantage is that this method is not designed to find anomalies and therefore may not work very well on all data sets.

The second category is the one based on the following assumption: *Normal data instances lie close to their closest cluster centroid, while anomalies are far away from their closest cluster centroid.* For this technique there two steps to follow. First the data is clustered using a clustering algorithm. Second, the anomaly score is calculated based on the distance to its closest cluster centroid, this is the data point in the center of the cluster.

The third category is based on the following assumption: *Normal data instances belong to large and dense clusters, while anomalies either belong to small or sparse clusters.* Techniques based on the above assumption declare instances anomalous when they belong to clusters whose size and/or density is below a certain threshold.

The key difference between clustering based and nearest neighbour based techniques is that clustering evaluates a data instance with respect to the cluster it belongs to and nearest neighbour evaluates a data instance with respect to its local neighbourhood.

Advantages and disadvantages

+ Clustering based techniques can operate in an unsupervised mode

+ Such techniques can easily be adapted for complex data types, it can be done by picking a clustering algorithm that can handle that particular data type

+ The testing phase is fast because the number of clusters that a data instance need to be compared with is a small constant.

- Performance of clustering based techniques is highly dependent on the effectiveness of the clustering algorithm in capturing the cluster structure of normal instances

- Many techniques are not optimized for anomaly detection, they detect anomalies as a by-product of clustering

- Several clustering algorithms force every instance to be assigned to some cluster. This might lead to anomalies getting assigned to a large cluster, this way they are considered as normal instances by techniques that operate under the assumption that anomalies do not belong to any cluster

- Several clustering based techniques are effective only when the anomalies do not form significant clusters among themselves

- The computational complexity for clustering the data is often a bottleneck

2.2.4 Statistical Anomaly Detection Techniques

Statistical anomaly detection is based on the following assumption : *Normal data instances occur in high probability regions of a stochastic model, while anomalies occur in low probability regions of the stochastic model.* Statistical techniques fit a statistical model to the data and determine the probability whether a data instance is part of this model. Both parametric as well as non-parametric techniques have been applied to fit a statistical model.

Parametric Techniques

Parametric techniques assume that the normal data is generated by a parametric distribution. The anomaly score is the inverse of the probability density function. The anomaly can also be determined with a hypothesis test, if it is rejected it is named anomalous. Based on the type of distribution assumed, parametric techniques can be further categorized as follows:

Gaussian Model Based Such techniques assume a Gaussian distribution. The parameters are estimated using Maximum Likelihood Estimates. The distance of a data instance to the estimated mean is the anomaly score for that instance. A threshold is applied to determine whether the data instances are anomalous.

Regression Model Based Anomaly detection using regression has been extensively investigated for time-series data. The basic regression model consists of two steps. The first step is to fit a regression model to the data. In the second step, the residual is used as test instances to determine the anomaly score. The residual is the part of the instance which is not explained by the regression model.

Anomalies in the training sample can influence the regression model leading to wrong answers. The robust regression method should solve this. It hides the anomalies but is also able to find them, because the anomalies tend to have larger residuals from the robust fit.

Mixture of Parametric Distributions Based These techniques use a mixture of parametric statistical distributions to model the data. There are two subcategories to this method.

The first subcategory models the normal instances and anomalies as separate parametric distributions. In the testing phase the technique determines to which distribution the test instance belongs.

The second category models only the normal instances as a mixture of parametric distributions. A test instance which does not belong to any of the learnt models is declared to be anomalous.

Non-parametric Techniques

In this case the model structure is not defined a priori, but is instead determined from the given data. These techniques make fewer assumptions regarding the data. There are a few categories of these techniques:

Histogram Based The simplest technique is to use histograms to maintain a profile of the normal data. A basic technique consists of two steps. The first step is building a histogram based on the different values taken by features in the training data. In the second step, the techniques checks whether a test instance fall into one of the bins. If it does not, it is labelled anomalous. For multivariate data, a basic technique can be extended to construct attribute-wise histograms.

Kernel Function Based A non-parametric technique for probability density estimation is parzen windows estimation. This involves using kernel functions to approximate the actual density. This method is similar to the methods described above, the only difference is the density estimation used.

Advantages and disadvantages

+ Statistical techniques provide a statistically justifiable solution, if the assumptions of the underlying data distribution hold

+ The anomaly score is associated with the confidence interval which can be used as additional information

+ If the distribution estimation is robust, this technique can work in unsupervised setting without need for labelled data

- The techniques rely on the assumption that data is generated according to some distribution, in practice this is almost never the case.

- Choosing the best statistical method is not a simple task

- Histogram techniques are simple but lack the possibility to model the interactions between different attributes

2.2.5 Information Theoretic Anomaly Detection Techniques

This technique is based on the following assumption: *Anomalies in data induce irregularities in the information content of the data set.* A basic information theoretic technique can be seen as follows. Search in a dataset for the minimal subset of instances where the complexity of remaining set of instances minus the complexity of this subset is maximal. This methods requires a lot of computational time so there are multiple variations proposed to fix this.

Advantages and disadvantages

+ It can operate in an unsupervised setting

+ It does not make any assumptions about the underlying statistical distribution for the data

- The performance of these techniques is very dependent on the choice of the information theoretic measure

- Information theoretic techniques applied to sequences and spatial data sets rely on the size of the substructure, which is often non-trivial to obtain

- It is difficult to associate an anomaly score with a test instance using an information theoretic technique

2.2.6 Spectral Anomaly Detection Techniques

Spectral techniques try to find an approximation of the data using a combination of variables to capture the diversity of the data. It is based on the following assumption: *Data can be embedded into a lower dimensional subspace in which normal instances and anomalies appear significantly different.* The general approach is to the determine such a subspace in which the anomalous instances can easily be identified.

One of these techniques analyses the projection of each data instance along the principal components with low variance. A normal instance will have a low value for such projection while an anomalous instance has a high value.

Another techniques is applicable for usage in time series of graphs. At every time instance, the activity vector for the given graph is determined. All of these activity vector are seen together as a matrix. With this matrix and a new test graph the angle is calculated to determine the anomaly score.

Advantages and disadvantages

+ Spectral techniques automatically perform dimensionality reduction and hence are suitable for handling high dimensional data sets, they can therefore also be used for pre-processing

+ Spectral techniques can be used in an unsupervised setting

- Spectral techniques are useful only if the normal and anomalous instances are separable in the lower dimensional embedding of the data

- Spectral techniques typically have high computational complexity

2.2.7 Handling Contextual Anomalies

The above anomaly detection techniques mainly focus on detecting point anomalies. Here we will discuss anomaly detection techniques that handle contextual anomalies. The research on contextual anomalies is very limited compared to the research on point anomalies.

Reduction to Point Anomaly Detection Problem

One way of dealing with contextual anomalies is to see them as point anomalies within a context. This way you can apply a known point anomaly technique within that context. A basic reduction based technique follows two steps. First identify a context for each test instance using the contextual attributes. Secondly compute the anomaly score for the test instance within the context using a known point anomaly detection technique.

An example of the generic reduction based technique is for the application where identifying the context is not straightforward. The data is represented as $[x, y]$ so the anomaly score can be calculated as a conditional probability. Another example is in the field of cellphone fraud detection. In this data, one of the attributes is the user. Within the records of a user searches for anomalies are performed.

Utilizing the Structure in Data

A generic technique in this category is described as follows. A model is learnt from the training data which can predict the expected behaviour with respect to a given context. If the expected behaviour is significantly different from the observed behaviour, an anomaly is declared.

2.2.8 Handling Collective Anomalies

As said before, collective anomalies are a subset of instances that occur together as a set and for which this set is anomalous. This is more complicated than the point and contextual anomalies since you have to explore the whole structure of the data.

Handling Sequential Anomalies

In anomaly detection literature two types of sequences are covered. The first type is symbolic, such as sequence of operating system calls. The second type is continuous such as time series.

The anomaly detection problem for sequences can be defined in different ways and are discussed below.

Detecting anomalous sequences in a set of sequences The key challenges of this technique are that the sequences might not be equal of length. Also the test sequences may not be aligned with each other or with normal sequences. This problem is addressed in following approaches.

- Reduction to Point Anomaly Detection Problem

  Certain techniques assume that all sequences are of equal lengths and treat each other as a vector and apply point anomaly detection techniques.

  Other techniques solve the fact that they are not equal of length by transforming each sequence into a record of equal number of attributes. Then point anomaly detection can be applied again.

  Another technique uses a distance/similarity measure between two unequal sequences. With this a clustering algorithms can be used.

- Modelling Sequences

  Sequential association modelling has been used to generate sequential rules from sequences. A test input is then tested with this set of rules to check if it is anomalous. Markovian modelling of sequences has been the most popular approach in this category.

  There are many more techniques proposed such as Probabilistic Suffix Trees (compact representation of a variable order Markov chain) and Sparse Markov Trees (similar to PST but it allows wild card symbols within a path).

Detecting anomalous subsequences in a long sequence Such anomalous subsequences have also been referred to as discords. This problem formulation occurs in event and time-series data sets where the data is in the form of a long sequence and contains regions that are anomalous.

One of the key challenges of this technique is that the length of the sequence is generally not defined. And since the input sequence contains anomalous regions, it becomes challenging to create a robust model of normalcy for this technique.

Different techniques are proposes such as Window Comparison Anomaly Detection (extract subsequences out of a given sequence of continuous observations using a sliding window) and Maximum Entropy Markov Models (predict the most likely state sequence for a given observation sequence).

Determining if the frequency of a query pattern in a given sequence is anomalous w.r.t its expected frequency Such formulation of the anomaly detection problem is motivated from the case vs control type of data. The idea is to detect patterns whose occurrence in a given test data set (case) is different from its occurrence in a normal data set (control).

Handling Spatial Anomalies

There is only a very limited amount of research done in this domain.

There is technique proposed to detect regions in an image that are anomalous with respect to rest of the image. The authors make an assumption that each pixel belonging to an anomalous region of the image is also a contextual anomaly within its segment.

There are two problems in the domain with graphs. The first problem involves detecting anomalous subgraphs in a given large graph. The second problem involves detecting if a given sub-graph is an anomaly with respect to a large graph.

3 | Operational Use

3.1 Data

3.1.1 Unmanned aerial Vehicle

The term UaV is short for Unmanned aerial Vehicle, more popularly called a drone. It is part of the UaS group, which stands for Unmanned aerial Systems. An UaV is an aircraft without a human pilot aboard. It flies under control of a human on the ground or completely autonomous. In our project an UaV will be set to a certain position above an environment and landed by a human. It captures videos of the environment autonomously.

Figure 3.1: Model of the used UaV

3.1.2 Collecting Data

The first thing that comes to mind for this project are the interesting things that we can capture from the data, the features. A feature is a specific (set of) fact(s) you can extract from data. You can think of the track someone is walking or the direction of this track.

We can let an UaV fly above an area for a certain amount of time to capture data. This can be a few hours to a few weeks, this depends on the amount of data we need to collect before we can make a sufficient model. This UaV captures images of an environment with a range of approximately 5 km by 5 km. There is a team of image processors who made a system that can extract data from these videos.

From these images we can get the *coordinates* of the people walking in the environment. Since the camera is at a fairly far distance, this is the only thing that can be captured. It will not be possible to see more detailed movement of the objects, so it is not possible to see that someone is looking at their watch for example. An example of a data entry could be: <ID, type, latitude, longitude>.

Then there is also the *environment*. People will behave differently in certain area's. It would be useful to know whether they are in a market place, or at a normal street. This can be determined in three ways.

- With an expert: the environments are manually assigned beforehand
- With the data: the environment is classified based on common hotspots and similarities in the data
- Grid based: the environment is divided in a grid

Another important piece of data is *time*. The time of the day, the day of the week and maybe even the day of the year. While in the morning it might be usual that it is quiet in the streets, this is not the case in the middle of the day. You want to take this into consideration when capturing the behaviour.

3.2 Features

Determining the features is a very important part of this project. It is only possible to capture 'normal' behaviour if you can describe it with the right features. For example, the behaviour of a human is not only captured by the look on his face but also on the rest of the body language. If you would only capture his facial expressions, you might get some results but you would get much better results if you also take the body language into account.

In this case the data that we can capture is coming from the UaV. The most important data we can collect from the images are the coordinates from people and the time. It is possible to track a person until he or she enters a building. When they walk out again, it is impossible to tell that it is the same person, because you only have a small image from above. This is not sufficient to determine an individual person unfortunately.

The features that we can retrieve from data are roughly divided in two categories. The features that you can capture can be based on only the object or based on the frame or area. These are called object based features and frame based features respectively[16].

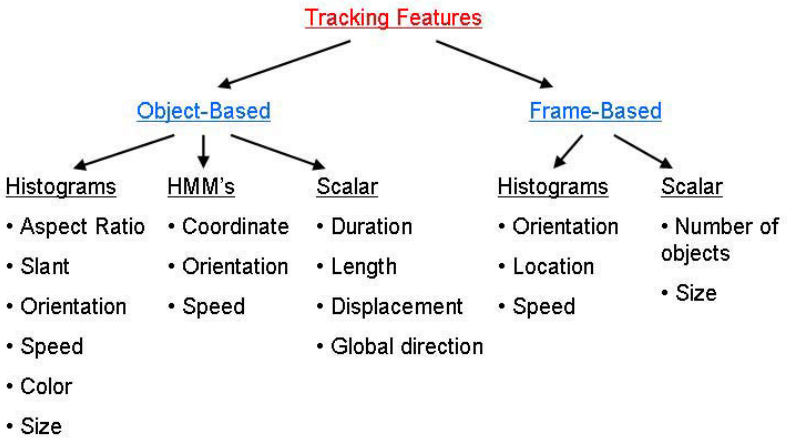


Figure 3.2: Possible set of features, taken from the paper of Porikli et al.[16]

3.2.1 Object Based Features

There are several features that describe an object in an environment. An obvious feature is the set of *coordinates* of an object. With these coordinates the track of an object can be determined. These tracks are important for the suspicious behaviour detection. If someone walks somewhere where normally no one walks, it is usually strange. You may also detect an object walking back and forth on the same track, this is also suspicious.

Another feature is the *lifetime* of an object. This is an important feature because the lifetime can be a good indication of strange behaviour. For example, if a person is on a certain square for a long time, while the rest of the people are passing by rapidly. This can be seen as unusual behaviour.

The *displacement* of an object over the total path. This can be seen as the place where he entered the environment till where he leaves. The same for the *global direction* which can be determined by the displacement.

Other information about this object would be whether the person is a male of a female. It might also be interesting to know whether it is a child or a adult.

3.2.2 Area/Frame Based Features

There are also several features that describe a frame. We will define frames here as a certain area in the environment and refer to it as area based from now on. These features are based on a certain area in the environment.

You can determine the *global direction* of all the object. This can be determined by the direction of the global displacement. This is interesting because on a street side where it is usual to walk from left to right, it is weird if a person suddenly walks from right to left.

Also the *speed* of all the objects. In an area where it is common to walk and not to run, such as a marketplace, it is strange when there is a person running.

Another feature is the *location* of all the objects. If in a normal situation everybody is on the left side on the area, it could be suspicious when there are everywhere at a certain moment in time.

There is also the *number of objects* in a certain area. It might be unusual to have a large amount of objects in a area where normally there are few, this indicates strange behaviour.

Then there is the *duration of objects* to end with. This is captured to locate people that stay abnormally long in a certain area.

3.2.3 Environmental Based Features

As we also discussed in section 3.1, there are also some important features that we want to take into account regarding the environment.

One of them is the *time*. This is important because you can think of the morning rush in a place that is much more quiet later on the day.

Another one is the *day of the week*. It could matter if you are analysing an environment on a weekend day or on a working day.

The *type of the object* could also be an interesting feature, if we are able to distinguish between humans and cars for example.

We could also add more features to the environmental based features, this is useful when knowing something about the environment and could improve the performance.

3.2.4 Feature Selection

In order to create a simple and elegant method, we can not keep track of all of these features, we therefore select the four most important features to ensure this goal.

The most important feature that we will focus on is the object based feature *coordinates*. We will capture the coordinates of the objects and use these to learn the trajectories of the objects.

The second feature that we will capture is the object based feature *lifetime*. We cannot capture strange behaviour of an object staying at a certain place for too long with just the trajectory. We therefore will capture the lifetime to ensure to notice this behaviour.

The third feature will be the area based feature *speed*. We want to know when an object is acting strange given a certain area. The speed of an area is not taken into account in only the trajectories. We therefore learn the normal speed for every assigned area to capture this strange behaviour.

The last monitored feature will be the environmental based features. We will monitor the time and day for this method. But the amount of environmental features can easily be extended. We only use these two because given more features we need more data to get to a good model, so we will start with a small set.

3.3 General Model

The procedure of our model will look like the diagram of figure 3.3. The UaV will fly above a certain area where it will capture data. This data is processed by an image processor which converts the data into coordinates. In this process each object will be processed and assigned own coordinates. For every object you get a trail of coordinates with a certain time stamp.

This data will be used for creating of model of 'normal' behaviour beforehand. Given this model we will provide an anomaly score for new entries afterwards.

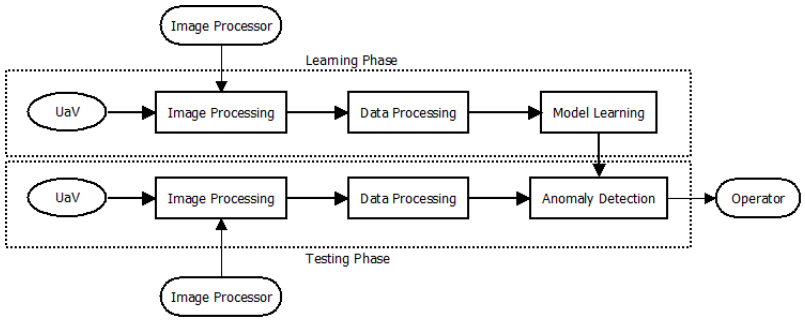


Figure 3.3: The general process of the model

3.3.1 Input and Output

Our method should be applicable on any new and unknown terrain. We therefore choose for an unsupervised learning method. This means that there is no predefined training data as the input. The training data that we can provide is the new incoming data and all entries in this data will be seen as normal. So there are no labelled instances of anomalous data instances. The rare instances that are anomalous during the training phase will be insignificant for the learning process given the rest of the data.

The output of the methods should be a level of anomalous behaviour of each object in the environment given to the security operator. The goal of the method is to assist the operator in his job. It will point out some possibly suspicious objects so the operator can take a look at them. This can be done for example by returning the top n anomalies to the operator or or in the form where the objects in the environment get a color coding that indicates the level of anomalous behaviour.

3.3.2 Learning and Testing

In the learning phase where we will learn the model of 'normal' behaviour we will pay attention to three aspects. We will learn the

- 'normal' paths of the objects in the area
- 'normal' lifetime of an object in the entire environment
- 'normal' speed of an object given a small area in the environment

With the 'normal' paths we can capture the trajectories of objects and also the direction. A path that is heading the other direction will be seen as another path. With the 'normal' lifetime we capture the objects that will stay of an extensive long period of time in the environment. With the 'normal' speed we capture objects that move extremely fast or slow. All of these features are caputured in combination with the environmental features.

In the testing phase we use the learned 'normal' behaviour to value a object as normal or anomalous.

4 | Method

4.1 Overview

In this chapter we will explain what kind of anomaly detection method we will use and how we came to this choice. Then we will explain how the clustering method that was chosen is applied to our situation, beginning with the simple single values. With some background of track clustering we come to the clustering method that we will use for the tracks which is more complicated than the single values. We will discuss what adjustments we needed to make to the existing method and how we implemented the different functions. We will also look into a suitable form to test new entries with the model that is learned with our method.

4.2 Method Choice

4.2.1 Requirements

To choose a method for our application we will first discuss the requirements that we need to take into account. As mentioned before, there will be no labelled data available. The method should be able to work with training data that is similar to the normal input. We also deal with a limit in memory use since it needs to work on a UaV where the capacity is not as large as in a normal computer. We also need to choose a method that facilitates fast anomaly detection because we need the detection to be real-time.

In summary, the requirements are:

1. No labelled training data required
2. Limited memory usage
3. Fast and on-line anomaly detection

4.2.2 Anomaly Detection Methods

We summarised the different methods for anomaly detection in section 2.2. Now we have to choose what kind of method is the best for our application. We will touch on each category and explain whether it is an option for our scenario. These findings are based on the information of the survey paper of Chandola et al.[4].

Classification based detection is a widely used method. For anomaly detection it suits well, it can detect anomalies fairly fast. For the learning phase however it is not applicable, classification based methods assume a certain knowledge of the different classes. This is not the case, we don't know what kind of area we are surveying and also nothing about the objects in the environment. We have no labelled training data to learn from, so classification based methods are not an option.

Statistical anomaly detection methods are also not usable. They assume an underlying pattern in the data that can be captured in a formula. This can only be found when we deal with data that is generated

according a certain pattern. This is difficult to determine, especially when we deal with patterns generated by humans. This method will therefore not be the best solution.

Spectral anomaly detection could be used in our situation. It can work in an unsupervised setting and is able to handle multidimensional sets of attributes. The disadvantage of this method is the high computational complexity, so it is a less preferable option for us.

The two large groups of anomaly detection methods that are left are the clustering and nearest neighbour methods. Both of these methods are able to work in an unsupervised fashion and are fast in the anomaly detection phase. In both of the methods you need to think about a suitable distance measure and their results are very dependent on this choice. The main difference between these methods is that the nearest neighbour method checks a data point to the local neighbourhood in contrast to the clustering method which checks it with regard to the cluster it belongs to.

4.2.3 Choice For Clustering

For our project we have chosen for a clustering method. We choose this over nearest neighbour methods because of the memory usage constraint. In a nearest neighbour method we need to remember the complete neighbourhood to check a new data instance. For a clustering method we only need to remember the learnt clusters which uses very little memory. This helps with the limited memory constraint. With clustering our model will consist of a number of clusters from which we only need to save the value of the cluster and the variance. All of these need just a small amount of memory which makes it perfect for our purpose.

As we mentioned before, the cluster methods can handle the fact that there will be no labelled data available. It builds the model of "normal" behaviour based on the input given, this input can be training data that consists of normal input. This will be translated in a certain amount of clusters that afterwards can be used for anomaly detection on test data.

The last constraint was the speed of the detection. This is very fast with clustering methods. Since we have a set of clusters that are represented by a value and a variance we can calculate the distances from an entry and the known clusters fairly fast. This is done with a defined formula that converts the distance to a value. The distance can be an actual distance in space but also a similarity measurement. The only challenge is to find a suitable distance measure for this purpose. With a suitable method this can even lead to real-time detection.

In particular we chose the method proposed in the paper of Piciarelli [14]. This method is cluster based and meets the requirements given. It will be explained in detail in the next sections.

4.3 Single Value Clustering

The first part of the data that we want to cluster are the simple values, the *speed* and the *lifetime* of an object. These two features are two important features as described in section 3.2. For both of these features we create a set of clusters which can be used during the detection part.

4.3.1 Distance Measure

To cluster the data we need a distance measure to calculate if a data entry belongs to a certain cluster. This can be calculated by the actual distance in space, for example with the euclidean distance. This may not give the best result because a difference of 2 km/hour might be close to a mean of 50 km/hour, but it might not be so close to a mean of 3 km/hour. We therefore use the variance of an cluster in the distance measure. This is the value that represents the size of the neighbourhood of a cluster that can still be considered as close.

We therefore use the euclidean distance measure with respect to the known variation:

$$d(t_i, c_j) = \frac{distance(t_i, c_j)}{\sigma_j} \tag{4.1}$$

where the distance is the euclidean distance:

$$distance(t_i, c_j) = \sqrt{(t_i - c_j)^2} \tag{4.2}$$

With this simple measure we can easily calculate the distance between a cluster and a data entry. If a data entry falls outside of all the known clusters (or when there are none) we will create a new cluster with a certain start variance. A cluster will be of the form: $(value, \sigma^2)$.

4.3.2 Update Cluster

If the entry matches a cluster it will be updated. It matches a cluster if it is in range of a predefined distance from the cluster, for example twice the variance. The updating of the clusters can be done in multiple manners. We can store all the information of the entries that match a cluster and add the new matched entry, then take the average and determine the variance. This would lead to very accurate clusters but the downside to this is the enormous amount of data needed to store all this information. Every time step there will be a very large amount of data added to this information. This is a problem for our specific scenario. We therefore choose for an approximation method. We store the value of the cluster and the variation and update these values with a ratio α of the new value. This is taken from the paper of Piciarelli [14]. This results in less accurate clusters but it gives beside the minimal memory usage, the extra advantage of adaption to changes over time. Updates from the past become less important with every update.

Updating of a cluster is done in the following fashion:

$$\begin{aligned} value_{cluster} &= (1 - \alpha)\ value_{cluster} + \alpha\ value_{new} \\ \sigma^2_{cluster} &= (1 - \alpha)\ \sigma^2_{cluster} + \alpha\ [distance(t_i, c_j)]^2 \end{aligned} \tag{4.3}$$

In this formula we again take the euclidean distance as the distance measure. The updating in this fashion also has an advantage if you will continue learning during the detection phase, since it will adapt faster to new situations.

4.4 Track Clustering Background

We want to learn the 'normal' paths of the environment with a cluster method given only the trajectories. This is mostly done in three steps, as described in Morris et al.[13]. The steps can be seen in figure 4.1. It begins with a preprocessing step where the tracks are prepared for the clustering step. The clustering step gives a compact representation of a path so it becomes easier to model these paths. Even though it is presented as three individual steps here, in practice learning steps are often performed simultaneously.

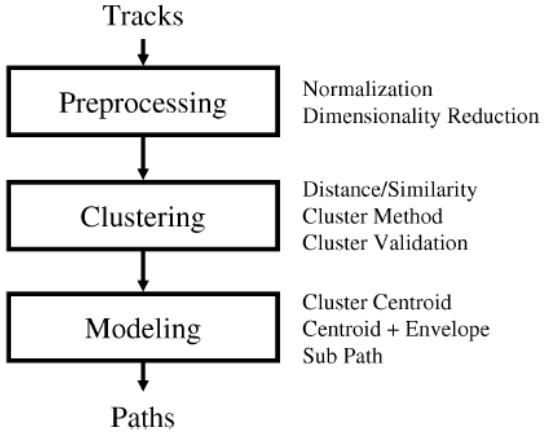


Figure 4.1: The general trajectory learning steps [13]

4.4.1 Preprocessing

In the preprocessing step the data will be transformed so it is ready for the clustering step. There the work will be done to enable a meaningful comparison between the tracks. This is useful because it allows us to use standard clustering techniques. This work is mostly done with a combination of normalisation and dimensionality reduction.

Normalization

Normalization of the tracks ensures that all trajectories have the same length. Two simple techniques for length normalization are zero padding and track extension. Zero padding places empty observations at the end of a track to make all tracks equally long. Track extension uses the information of the latest tracked points to estimate the next point as if they were observed from that time. Both of these techniques extend the shorter tracks to tracks of the size of the longest track in the training set. These methods lead to a very large data set of trajectories.

Normalisation methods only work on complete tracks, it is not possible to use these techniques in an on-line system. For on-line anomaly detection we need to directly analyse the tracks, not only afterwards. The benefits of these methods are that they are computationally simple and keep the trajectories intuitively understandable.

Dimensionality Reduction

Dimensionality reduction is applied to create a space that is easier to use for computations. It also creates a space for more robust clustering since it reduces the training set data. This is in our case not a problem because we have data in a very compact form.

Dimensionality reduction avoids the curse of dimensionality which creates more robust clustering, but if the chosen model does not represent the tracking process well, the resulting clusters will be meaningless.

The method proposed by Piciarelli [14] is able to deal with unfinished tracks. This makes it unnecessary to apply any normalization technique. As mentioned, dimensionality reduction is also not needed. This means that we can used the data as we receive it, which saves a lot of time.

4.4.2 Clustering

To identify structure in data we use clustering. When observing an environment, we collect motion trajectories and they are grouped into similar categories we call paths. We want to get meaningful paths so we have to look into three important issues.

Distance/Similarity

The first issue is the definition of a distance or similarity measure. Most clustering techniques strongly depend on the definition of distance or similarity. The main difficulty with tracks is the potential unequal length even though they might be produced by the same event. This can be solved by preprocessing but we also need to choose the distance measure wisely.

One of the most known distances measures is the Euclidean distance. For this method the two tracks that we want to compare must be of the same length. There is however a version that solves this issue, here the tracks can be of different lengths. It is a simple

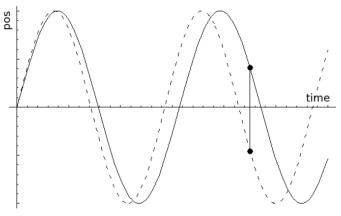


Figure 4.2: Euclidean distance is not suitable for time series [14]

measure and it performs poorly in combination with time shifts as you can see in figure 4.2.

Another distance measure has been developed by Piciarelli et al. [14] that does not require a whole trajectory. This distance measure takes the average of the minimum distance between two tracks in a sliding window. The minimum distance is based on the best match for a track.

There is also the Hausdorff distance which measures the distance between two unordered sets. This method can be used for tracks of unequal length but it does not take into account the ordering which can lead to incorrectly labelled similar tracks. For example a sidewalk where people walk in two directions is labelled as one track.

These are all examples of using the distance as a measure, but you can also use a similarity value. This value is based on the distance measure combined with a parameter to control how quickly similarity drops with increasing distance. This way you have an extra parameter to tune for performance, this can be an advantage since the measure can be better adjusted. It can also be a disadvantage since it can be hard to tune it correctly.

Clustering Procedures

The trajectories can be grouped using unsupervised learning techniques once they have been preprocessed. They are grouped together in clusters that are perceptually equal, these clusters are called routes. The clustering can be done with different techniques.

We describe the advantages and disadvantages of the most common ones. These techniques are taken from the paper of Morris et al.[13].

- *Iterative Optimization* is very simple but quite effective generally. A downside is that the number of clusters must be determined manually and the data must be of equal length.
- *On-line Adaption* is well suited for real time application because the number of clusters is not specified and it adapts to changes. However, it may be difficult to specify correct criteria for the cluster initialisation that does not include the outliers and there are no guarantees about optimality.
- *Hierarchical Methods* are well applicable for graph theoretic techniques and it allows for an intelligent choice of the number of clusters. But the decisions on how to split a set highly influences the quality of the clusters.
- *Neural Networks* can represent complex non-linear relations in a low-dimensional structure and it can be trained to be updated when there are unseen examples. When there are large networks it requires are lot of data and may need a very long time to converge. It can also be difficult to learn the correct weights and parameters from the data.
- *Co-Occurrence Methods* uses a model where the trajectories are viewed as bags of words and the routes as similar bags contain similar words. This means that the technique is independent from the length of trajectories. A disadvantage is that the vocabulary size can be limited and therefore the clusters are not optimal. Also time-ordering is mostly not preserved.

Cluster Validation

The quality of the paths that have been clustered needs to be determined because the true number of clusters is always unknown. This is especially the case when the used techniques require a predefined number of clusters. An example of a criterion that can be used to determine the quality of a clustering is the Tightness and Separation Criterion (TSC) which measures how close trajectories in clusters are compared to the distance between the clusters. Another example comes from the information theory, this is the Bayesian Information Criterion (BIC).

For the distance measure we use the formula from Piciarelli et al. [14], this will be further explained in the next section. As the clustering procedure we use the on-line adaption, this fits best with our requirements. It allows us to use the method real-time.

4.4.3 Modelling

The resulting paths are modelled when the trajectories have been clustered. The paths can be modelled in two manners. The first considers a path as a whole, from start to end point, this is called a full path model. The second decomposes a path into smaller parts called subpaths, this is called a tree of subpaths. See also Figure 4.3.

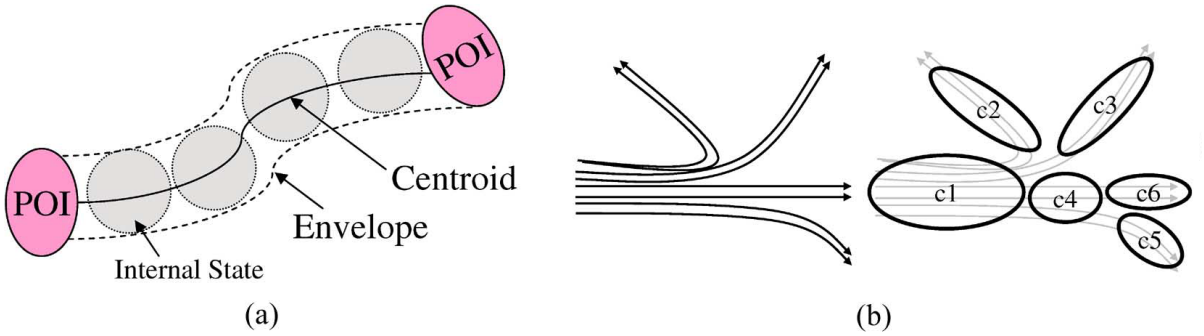


Figure 4.3: Representation of a full path model (a) and tree of subpaths (b) [13]

A path in a full path model is specified by its centroid, this is the backbone of the path. It is the average trajectory of a cluster. The path is further specified by extending the centroid with an envelope. The envelope is a way to express the variation of a path, it is the space in which a point is still considered to be a part of this path.

In the representation as a tree of subpaths, the path is divided into smaller parts. These parts represent similar regions of a path. These regions are connected in a way that they represent the transitions between subpaths.

4.5 Clustering Method

In this thesis we will use an adaption of a method proposed by Piciarelli et al. [14]. We use this method because it directly corresponds with our requirements. To recall the steps for track clustering:

- *Preprocessing*: The method of Piciarelli is able to use unfinished trajectories. We therefore do not need to normalize the tracks which is a huge advantage, otherwise we would not be able to track an object while walking in the environment but only afterwards. As mentioned before we do not need to reduce the dimensionality of our data.
- *Clustering*: For the clustering the method of Piciarelli uses its own distance measure where we do not need a complete track to be able to compare it. It also uses an on-line adaption clustering method. This means that the clusters are updated during the execution of the method and can be used directly. This results in a fast algorithm and this is useful when we process a large amount of data.
- *Modelling*: The representation of this method is as a tree of subpaths. We can easily use these subpaths for the calculation of probabilities of certain path changes.

4.5.1 Definitions

Before we explain the clustering algorithm we will first give some definitions.

We have a set of trajectories T, a trajectory T_i of an object is of the form:

$$T_i = \{t_{i1} \dots t_{in}\} \text{ where } t_{ij} = \{x_{ij}, y_{ij}\} \tag{4.4}$$

where a trajectory T_i consists of vectors t_{ij} representing the positions x and y at time step j of object i. This time is the order of the trajectory, this is not the real-time.

We have a set of trajectory clusters C, a trajectory cluster C_i will be of the form:

$$C_i = \{c_{i1} \dots c_{im}\} \text{ where } c_{ij} = \{x_{ij}, y_{ij}, \sigma_{ij}^2\} \tag{4.5}$$

where a cluster C_i consists of vectors c_{ij} where x and y are positions at time step j and σ_{ij}^2 is an approximation of the local variance of the cluster i at step j.

4.5.2 Distance Measure

When a new trajectory is detected, a check is done to see if it matches an existing cluster. Therefore we need a distance measure to check if a trajectory matches a given cluster. The measure is defined as follows: given a trajectory $T = \{t_1 \dots t_n\}$ and a cluster $C = \{c_1 \dots c_m\}$:

$$D(T, C) = \frac{1}{n} \sum_{i=1}^{n} d(t_i, C) \tag{4.6}$$

where

$$d(t_i, C) = \min_j \left(\frac{dist(t_i, c_j)}{\sqrt{\sigma_j^2}} \right) j \in \{\lfloor (1-\delta)i \rfloor \dots \lceil (1+\delta)i \rceil\} \tag{4.7}$$

where $dist(t_i, c_j)$ is the Euclidean Distance:

$$dist(t_i, c_j) = \sqrt{(t_i - c_j)^2}$$

This distance measure is the mean of the distances to the cluster C. The closest cluster is determined by the 'normal' euclidean distance within a sliding window centered around time step j. This window is determined by δ on two sides dependent on the progress of the track. The further the track is developed, the larger the range will be. This distance is normalized by the variance. The window is clipped if it falls partly outside the cluster and the distance is set to infinity if the window falls completely outside a cluster.

4.5.3 Cluster Updating and Creation

If a trajectory does not match a cluster, a new cluster will be created. This is done by taking the x and y values of the current position of this trajectory and the starting σ^2, which is a predefined value.

When the trajectory matches a cluster, this cluster needs to be updated in the following fashion.

If $c_j = \{x, y, \sigma^2\}$ is nearest to the trajectory element $t_i = \{\hat{x}, \hat{y}\}$ then c_j is updated as follows:

$$\begin{aligned} x &= (1-\alpha)x + \alpha\hat{x} \\ y &= (1-\alpha)y + \alpha\hat{y} \\ \sigma^2 &= (1-\alpha)\sigma^2 + \alpha[dist(t_i, c_j)]^2 \end{aligned} \tag{4.8}$$

Here $\alpha \in [0, 1]$ is an update rate for the clusters. This is useful for when the situations in the past are less relevant then the latest situations and when there is a limited memory.

4.5.4 Tree of Clusters

We want to build a tree of clusters with this algorithm that we can use for anomaly detection. The tree is structured in such a manner that it can be used for probabilistic reasoning. A trajectory is not represented as a single cluster but as a path in the tree consisting of multiple clusters.

The tree building is globally done as seen in Figure 4.4. For every new point on a trajectory its distance is measured to each starting cluster. If there is no starting cluster near enough, there is no matching starting cluster and a new starting cluster is created. This means that a new tree of clusters is created with only the root node.

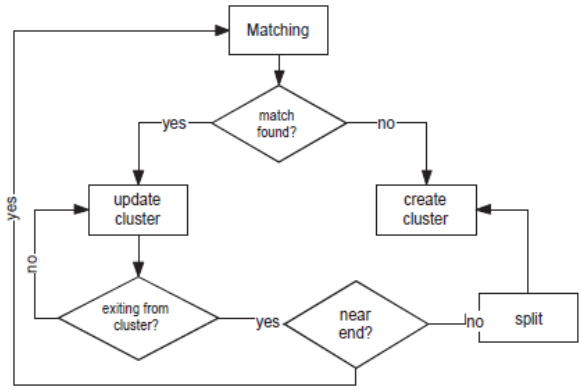


Figure 4.4: The tree building process [14]

In case there is a starting cluster near enough, a match is found. This nearest cluster is updated with this new point. As the cluster is updated, the distance from t_i to this cluster is monitored to see when it moves away from the cluster. If the space between the track and the cluster becomes too large, the trajectory is no longer matching the cluster at a certain distance.

This can be the case when it exceeds the cluster or it is exiting at a point far away from the end of the cluster. An example can be found in figure 4.5. When a track is exiting a cluster at the end of the cluster (see Figure 4.5.3), the new match is searched among the children. If a match is found it continues with this child cluster (see Figure 4.5.4). When a track is leaving a cluster but it is not near the end (see Figure 4.5.5), we will split the current cluster and create a new cluster with the previously walked cluster as parent (see Figure 4.5.6).

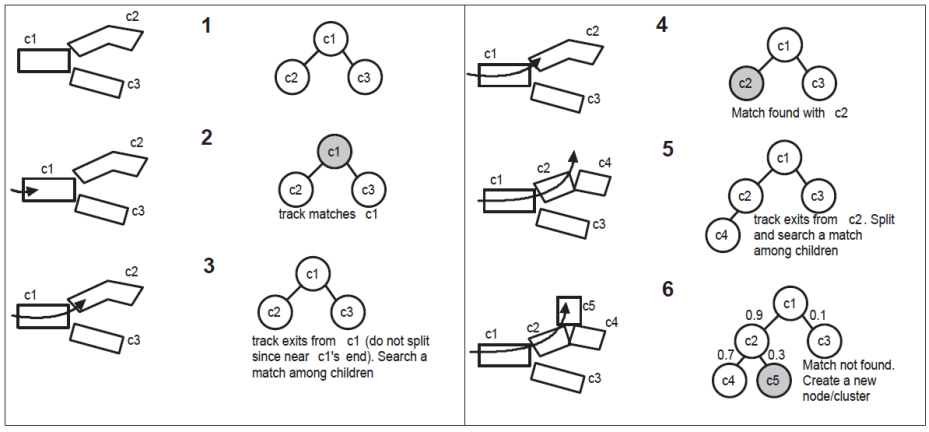


Figure 4.5: An example of the the creation process [14]

4.6 Adjustments

To improve the method used by Piciarelli we made some adjustments, we will discuss these in the following sections.

4.6.1 Ambiguity of the Method

During the implementation of the method we found a few sections in the method that were not completely clear.

Adding points to a cluster

The first problem we came across was the forming of the clusters. In the paper was mentioned that one cluster consists of multiple points but nowhere was mentioned when points are added to a cluster, only when points are updated.

We choose to interpret it as follows: when a new cluster is created all of the points that follow are added to this cluster. For example when the method starts there are no known paths, this will result in a new cluster. All the coordinates of the object from that point on are added to this new cluster. When another track of an object is matched to this cluster it is updated.

Distance measure

We use the distance measure as discussed in section 4.5.2. It is explained that the distances between a cluster and a path is the average of all the closest distances in a certain range. If we implement this measure literally we come across the following issue.

When a path is leaving a known path, even perpendicular, the distance between this path and the known cluster will stay quite close for a while. The result of this is a strange offset in the known cluster that will occur. This happens because as long as the path is 'close' enough to the cluster, the closest point is updated. As seen in figure 4.6, this will result in a non-logical offset with the extra disadvantage that the variance will become quite high.

Therefore we made an assumption about the distance measure that was not clear in the paper. When we calculate the closest distance from a point to a point in the cluster we only use this distance when it is below a certain threshold. Otherwise this distance will be set to infinity. In figure 4.6 we can see that the paths have some strange paths even tough the path itself is perfectly straight. When we apply this improvement we get the clusters as shown in figure 4.7. The tracks will not influence the clusters when they have a clear path in another direction away from the cluster. This means that the tracks in the improved version are not as long dependant on the previous course of their track than the original method is. This results is more straight and logical lines as we can see in figure 4.7.

4.7 Implementation and Design Choices

4.7.1 Calculate Distance

For the calculation of the distance we use the distance measure as described before in section 4.5.2. It works as follows, for every step in the path of the object we will look for the closest point in the cluster. This point is searched in a window around the step in the path. If the closest distance is too far away it becomes infinite. When all of the steps are calculated we take the average of these distances, this is our value for the overall distance.

We chose to set the values of the sliding window to 0.5 and 1.5. We chose this large window because it assures also in the beginning of the path a solid value for the distance. We chose to set the minimal

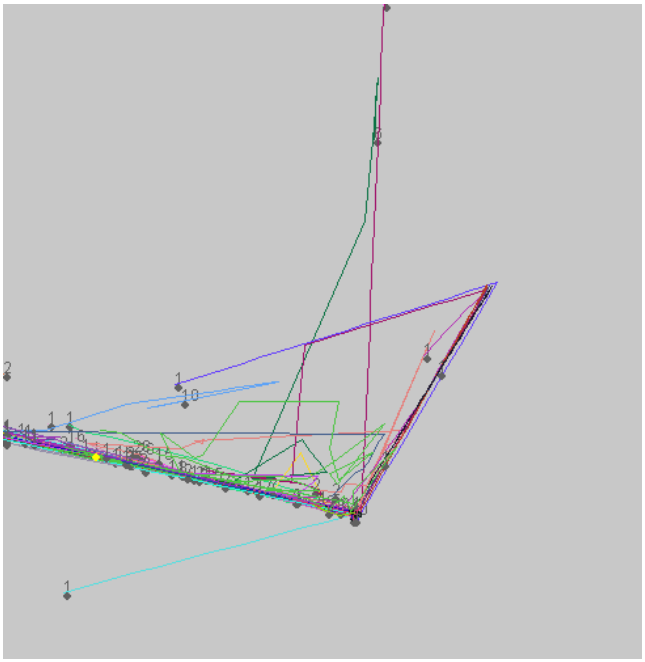


Figure 4.6: Before the adjustment

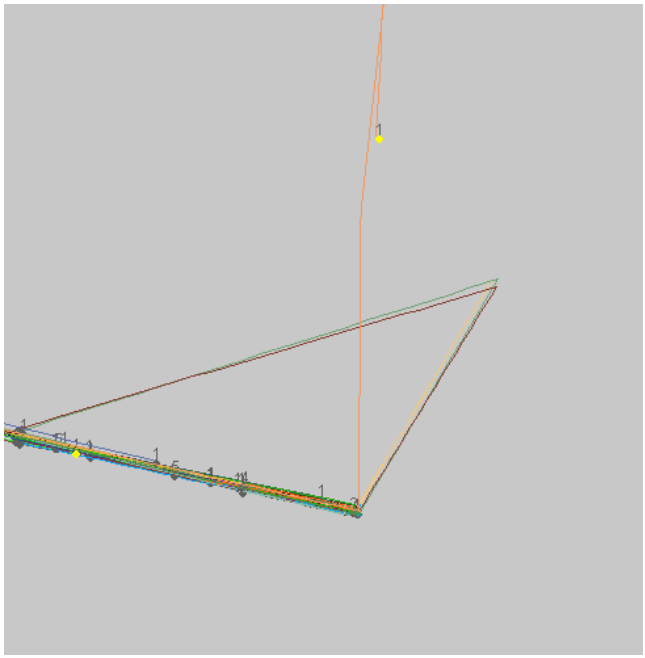


Figure 4.7: After the adjustment

distance to be taken into account to 4 times the variance of that certain point in the cluster, because in that way a lot of points are taken into account but the really far points are ignored.

Algorithm 1 Calculates the distance between a track and a cluster

```
function CalculateDistance(path, cluster)
    mindistance = 0;
    for i ← 0 to size(Path) do
        mindistance = ∞;
        beginindex = i * 0.5;
        endindex = i * 1.5;
        for j ← beginindex to endindex do
            if j is inside scope cluster then
                tempmin = euclideandistance(cluster(index j), track(index i))/variance(cluster(index j));
                if tempmin < 4 * variance(cluster(index j)) then
                    mindistance = tempmin;
        subdistances = subdistances + mindistance;
    totaldistance = 1/size(Path) * subdistances;
    return totaldistance
```

We start by setting the minimal distance to 0. For every point in the path we will determine the local distance. We do this by looking in a window between 0.5 and 1.5 times the step in the path. For example at step 4 in the path we will look between 2 and 6. For every corresponding step in the assigned cluster, the distance is determined by the Euclidean distance divided by the variance. If this distance fall in the range of 4 times the variance at that point in the cluster, it is considered as a valid distance. From all these distances the smallest is taken. This is done for every step in the path and the average of all these distances together is the return value.

4.7.2 Update Cluster

Once the previously calculated distance is close enough, the cluster needs to be updated. This is done as described in section 4.5.3. The point that we will update is the point in the cluster closest to the last step of the path.

For the value of α we chose 0.05. This means that the values are updated with 5% of the new value in every update. This rate makes sure that the cluster is not changing too much with every update, but every update do makes a little difference. We also added an extra constraint that the value of the variance can not exceed the starting variance.

Algorithm 2 Update a cluster

```
function UpdateCluster(object, cluster, index)
    x = (1-α)*clusterX(index) + α * objectLocationX;
    y = (1-α)*clusterY(index) + α * objectLocationY;
    variance = clusterVar(index)
    if cluster != root then
        variance = (1-α)*clusterVar(index) + α*eucldist(cluster(index), track(index))/var(cluster(index));
    if variance > maxvariance then
        variance = maxvariance;
    cluster(index) = (x, y, variance);
```

When we update a cluster, the x and y values are updated with 5% of the new value. The value for the variance is more complicated. If we are updating the root of a tree, then we leave the variance the same. This is to make sure that new entries are able to match to this tree. For any other cluster, we update it with 5% of the distance between the track and the cluster. If the new variance exceeds the maximal variance set then it is set back to the maximal value allowed.

4.7.3 Split Clusters

When a track does not match a cluster anymore and it does not leave a cluster at the end, the cluster needs to be split. For the split we calculate the index of the split as the point in the cluster closest to the last step of the path that is leaving the cluster. This method is not described in detail in the paper.

When we split the coordinates, we take the sublist from 0 to index +1 to make sure there will be no gap between the two clusters that are split.

Algorithm 3 Split a cluster

```
function SplitCluster(cluster, index)
    if index > 0 then
        newcluster = cluster;
        tracks(newcluster) = subtracks(cluster, index to end);
        tracks(cluster) = subtracks(cluster, 0 to index + 1);
        for cluster c:children(cluster) do
            newcluster.addChild(c);
        cluster.deleteChildren();
        cluster.addChild(newcluster);
```

We will split a cluster when it has an index higher then 0, this is to prevent leaving an empty cluster. We first make a copy of the current cluster. For this new cluster we remove the tracks before the index point and for the old cluster all the tracks after the index point. Afterwards we need to correctly change the parents of these clusters.

4.7.4 Merge Clusters

When a track is completed we will check if some tracks of the tree are so much alike that they should be merged together. The only constraint known from the paper is that two clusters need to be on the same level in the tree to be merged together. For every possible pair of two clusters we check if they are close enough. We go along the track until the two points are too far apart, this index is used in the algorithm.

Algorithm 4 Merge two clusters

```
function MergeClusters(cluster1, cluster2, index)
    for i ← 0 to index do
        x = occ(cluster1)/occ(total) * cluster1X(i) + occ(cluster2)/occ(total) * cluster2X(i);
        y = occ(cluster1)/occ(total) * cluster1Y(i) + occ(cluster2)/occ(total) * cluster2Y(i);
        variance = occ(cluster1)/occ(total) * cluster1Var(i) + occ(cluster2)/occ(total) * cluster2Var(i);
        cluster1 = (x, y, variance);
    tracks(cluster2) = subtracks(cluster, index to end);
    cluster2.setParent(cluster1);
    if size(tracks(cluster1)) - index > 1 then
        splitCluster(cluster1, index);
```

When we merge two clusters we take the average off the x, y and variance values based on the occurrence of these values in the clusters. We save these values in $cluster_1$ and afterwards we remove these points from $cluster_2$. Then we set the parent relation to the correct values. If there are remains of the first cluster, we split these and create a new cluster of these points.

4.7.5 Concatenate Clusters

We use a recursive function to concatenate all the clusters in a tree when a cluster only has one child. This can occur when clusters are merged together. This recursive function ends when a cluster has no more children, or if it is at a leaf of the tree.

Algorithm 5 Concatenate two clusters

```
function ConcatenateClusters(cluster)
    children = cluster.getChildren();
    if size(children) == 0 then
        return;
    else
        if size(children) == 1 then
            child = children(0);
            for cluster c:children(child) do
                cluster.addChild(c);
                c.setParent(cluster);
            cluster.removeChild(child);
            if size(tracks(child)) > 1 then
                cluster.addTracks(tracks(child));
    for cluster s:children(cluster) do
        concatenateSubtrackClusters(s);
```

If a cluster has children, this function will try to concatenate clusters. If it has exactly one child, we can concatenate these clusters. For every child of the child we set this as a child of the parent. Then we add the tracks of the child to the parent and remove the child. This function is repeated for all children and so on.

4.8 Testing Phase

In order to come up with a value for an object which determines the anomaly value we need to take a few things into account. We can determine the 'anomaly value' in multiple ways. We need to create a formula that incorporates the three features that we monitor. We therefore need to create an 'anomaly score' for the speed, the lifetime and the paths taken.

4.8.1 Speed and lifetime

For the speed and the lifetime we deal with a set of single value clusters. It is easy and fast to calculate the distance to the closest cluster. The distance measure used will again be the Euclidean distance in combination with the variance. This will be the first feature of the anomaly score, the second will be determined by the environmental attributes. For this measure we take the ratio in which this closest cluster occurs within a time/day frame compared to the whole. This ratio gives an indication of the likelihood of a clusters given the time and day.

4.8.2 Paths of objects

For the path it will be nearly identical to the single value clusters. At a certain point in time it is possible to calculate the distance to the currently assigned closest cluster. The distance will be calculated in the same fashion as the distance is calculated in the learning phase, see equation 4.6. Also for the paths we will determine the likelihood of a path given the environmental attributes. We will do this in the same fashion as described above.

4.8.3 Formula

In summary we have the following measurements:

- Speed distance : distance(object, closest cluster)
- Speed environment: P(cluster x | day & time)
- Lifetime distance : distance(object, closest cluster)
- Lifetime environment: P(cluster x | day & time)
- Path distance : distance(object, closest cluster)
- Path environment: P(cluster x | day & time)

Combining these features in one anomaly score can be done in multiple ways. One obvious option is as follows:

$$speed(distance * 1/probability) + lifetime(distance * 1/probability) + path(distance * 1/probability)$$

but to improve on this we can assign weights to the different features, some may prove to be more important. Then we will get:

$$w_1 * speed(distance * 1/probability) + w_2 * lifetime(distance * 1/probability) + w_3 * path(distance * 1/probability)$$

But of course there are many more possibilities.

4.8.4 Evaluation Testing

Unfortunately we had no time to compare the possible formula's. We can therefore only guess which one will work best in practice. An extensive test of these formula's can be done as future work.

5 | Additions

During the testing a few of the disadvantages of the used model came forward. We will discuss those and offer some possible solutions and improvements. In this section we also discuss the additions that we made to the method.

5.1 Environment

A new adjustment is the use of the environmental information. We use this information on all of the aspects that we capture during the learning phase. The environmental information can consist of different aspects such as the type of object. In this project we limit the environmental information to the day of the week (7 values) and the hour of the day (24 values). This results in a table of 7 by 24 for the different values. This matrix is easily extended with new aspects, it requires only a higher dimensional matrix.

	Monday	Tuesday	...	Sunday
0:00	#	#	...	#
1:00	#	#	...	#
⋮	⋮	⋮		⋮
22:00	#	#	...	#
23:00	#	#	...	#

Figure 5.1: The environmental information

One option was to build a separate model for each cell in the table. This would give a very specific model for each different time zone of the week. However this model would need a lot of data to create a fitting model. We therefore choose to create a model that does not take these values into account when creating the model. Every time a cluster is updated the frequency of that particular table cell is increased so it can be used in the anomaly detection phase afterwards.

For example when a speed of 40 km/h occurs on Monday at 10:00, it will add an occurrence to the cell in the table corresponding to Monday and 10:00. When we build the table in this manner, we can subtract a likeliness of a path in the model given a certain time and day. We do this by taking the probability of a certain speed for example given a time stamp. When a speed of 10 km/hour only occurs in the evening, this speed will get a low likeliness in the morning.

5.2 Problem: Specific Trees

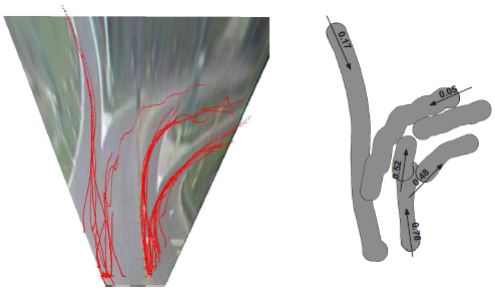


Figure 5.2: Example of the testing environment of Piciarelli in his paper

The method in the paper [14] was developed for anomaly detection of moving objects and that is exactly what we want to achieve in this thesis. Unfortunately the paper had only tested the method on a small scale. They considered only simple roads such as a highway on a small section.

In such environments the starting points are limited, only at the beginning of the roads and at the edge of the view. The starting point is chosen when an object is noticed in the environment. When noticed we investigate if there is a starting point close enough, if not we create a new tree with its starting point as the root of the tree.

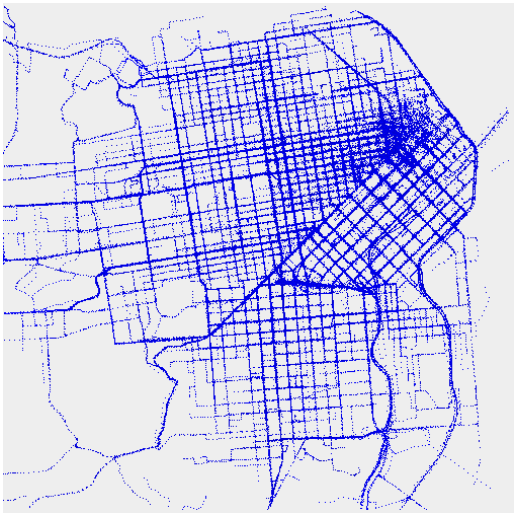


Figure 5.3: Map of San Francesco

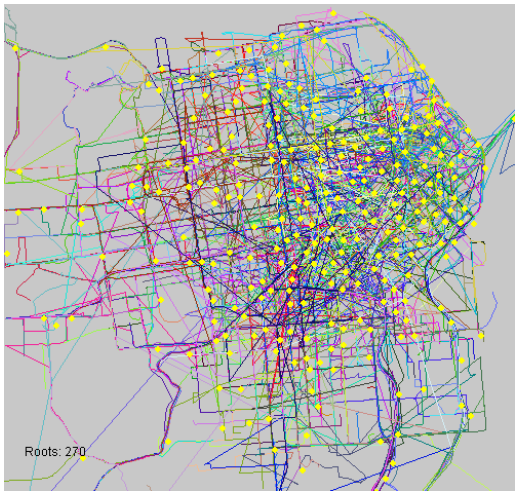


Figure 5.4: Map with starting points

We take a larger environment such as San Francisco for example to test the method on [1]. We can see that in this setting there are many starting points and this is logical since it is a large environment, all of the residents can create a starting point. A big downside to this is the amount of data needed to create a good model for each of these starting points. From each starting point a new tree is built and every tree needs a certain amount of data.

This model makes the clusters too specific. Because of all the starting points we get a lot of trees which have the possibilities to follow the same path when crossing the highway for example. Since it is not possible to merge two trees somewhere in the middle, since it would remove the characteristics of a tree, it would create a graph. We could get a lot of overlap in the trees which is not recognised in the model. The complete map and the model with the starting points are respectively found in Figure 5.3 and 5.4. Two examples of a tree in the model can be found in Figure 5.5. As you can see, even though the complete map is full of paths, the number of paths in one tree are quite small.

Figure 5.5: Two examples of trees in the origional method

5.3 Solution: Starting Grid

We want to create less starting points to assure that those starting points contain more data. We will try to set the root of the tree from the moment that the smaller routes are already behind them. We will do this by ignoring the first part of the route taken. We determine a grid of a certain size in the environment and we start following an object as soon as it crosses one of the boundaries between the grids. We therefore only create tree roots on the intersections of the grids. This will result in less trees and also in less useless

Figure 5.6: Two examples of trees in the adjusted method with a grid of 5 by 5

[1] http://www.infochimps.com/datasets/uber-anonymized-gps-logs

information. The idea behind creating a staring grid
is that we lose the information at the beginning of the tracks of the objects, but that this part is the least interesting. It is more interesting to see where an object is heading as soon as it has been on the road for a while and already put the small beginning paths behind it.

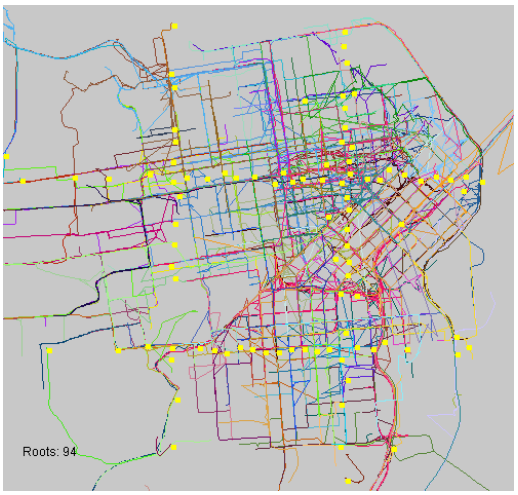


Figure 5.7: Map with starting points on grid of 3 by 3

Figure 5.8: Map with starting points on grid of 5 by 5

5.4 Problem: Time Dependence

Another problem that we detected using this method is that it is highly time dependent. This means that in the clusters of the tracks the time dependence is implicitly taken into account. If all the objects in an environment would walk or drive at the same speed, this would not be a problem. However in all practical applications this is not the case.

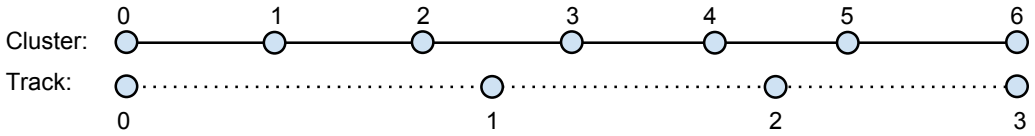


Figure 5.9: Example of the time dependence

Take for example the situation of Figure 5.9, a cluster with 7 points in it. When we take a look at the track in this figure it is clear that it is moving exactly the same path (assume it is on the same line) but according to this method it does not belong to this cluster anymore at a certain point in time. Following the algorithm we know that we will compare the point in the track to the points in the cluster in a certain window. This window is based on the number of points observed in the track. We use this number to determine the window in the cluster, so when we do this for this situation.

t=0 Look in the cluster from point 0 to point 0
The distance between Cluster(0) and Track(0) is small

t=1 Look in the cluster from point 0 to point 2
The distance between Cluster(2) and Track(1) is small

t=2 Look in the cluster from point 0 to point 3
The distance between Cluster(4) and Track(2) is small

t=3 Look in the cluster from point 0 to point 5
The distance between Cluster(5) and Track(3) is not so small anymore

We can see that the further we get in the process the larger the distance will become and it won't take long before the algorithm will not recognise it as the same anymore. In this case probably at time step 3 or 4. And this is only an example where the speed is approximately twice as much. The difference between pedestrians and vehicles can be even greater.

5.5 Solution: Space Dependence

A solution to the time dependence is to take out the time constraint. We can do this for example by only registering a move if it is at least a number of meters further in the environment. We do this by taking the Euclidean distance from the last 'registered' position to the current position, if the distance is above a certain distance we will register this position, otherwise it will be ignored. We solve the time dependence problem with this because if two objects start at the same position and move in the same direction but at a different speed this will be no problem. The object that is moving slower will be 'noticed' less often and this would lead to more or less the same track.

An example of this is visualised in Figure 5.10. Since we will only register points after a certain travelled distance, original points 1,3 and 5 will be ignored. As we can see, the cluster is now much more usable for the track, which will be processed in the same fashion.

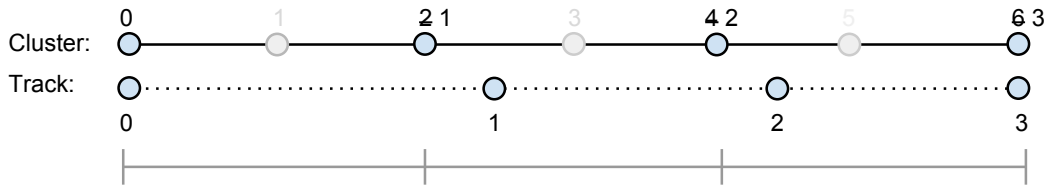


Figure 5.10: Example of the space dependence

5.6 Parameter Settings

Another disadvantage is that this method is very dependent on the choice of a few parameter values. Before we run the program we will have to manually set a few values. This is the case for the variance of the speed, lifetime and tracks and for the values of the grid and step size. The outcome of the model is very dependent of these settings. In a model that should work without supervision or prior knowledge it would be better if the settings are not that determinative for the outcome. Since they are in this case, we need to make a reasonable guess of these values that can only be done with trial and error or experience with the model.

The value for the grid size determines the number of trees in the model. The smaller the value for the grid, the smaller the amount of trees will be. This becomes clear in figures 5.7 and 5.8.

The step size determines how the objects are taken into account. If you set it too low, it will not have an effect since every step is registered but if you set it too high then some objects may have very little registered points. With few points it takes much longer to build a good model.

But the greatest dependence comes with the value of the initial variance.

5.6.1 Initial Variance

This method is highly dependent on the choice of the initial variance. This variance is used in all of the eight following functions in the method:

- addTrackVector: adds a new point to a cluster
- updateSubtrackCluster: the updating of a point in the cluster
- newSubtrackCluster: creating a new cluster
- checkChildOfCluster: checks if a cluster is possibly a child of the cluster
- splitCluster: splits two clusters at a given point
- checkSplitCluster: checks if a cluster can be split
- mergeSubtrackClusters: merges two clusters together
- checkCurrentCluster: check if a track still belongs to the assigned cluster

This means that a slight variation in the value of the variance can lead to enormous variation in the learnt model.

5.6.2 Variance to Zero or Infinity

Along the process the variance is updated a lot. It can be possible that in this updating process the variance can become very small, when all the tracks are approximately on the same place, or very large, when all tracks are very far from each other. The first can become a problem when all the tracks are exactly the same, then the variance can become 0 and no new track will be matched to it. This is a problem that will not very likely happen in a real environment. The second can become a problem when all tracks are very different, than the variance is updated every time to a higher number. When the variance becomes too high, almost everything can be matched to this track. This means that there will not be much detected as anomalous in the detection phase. We solution to this problem very simple, we set a boundary to the variance. Whenever the variance exceeds the max variance, it is set back to the max set value.

6 | Results

6.1 Test Case

In order to show that the method is produces the correct results for a simple data set, we created a data set of our own. We chose a very simple case with only one starting point but with all the possible curves to check if the spitting and updating mechanisms are working correctly. Figure 6.1 shows the paths in this data set and figure 6.2 shows the clusters that should be returned by the method. The clusters are the black boxes in this figure.

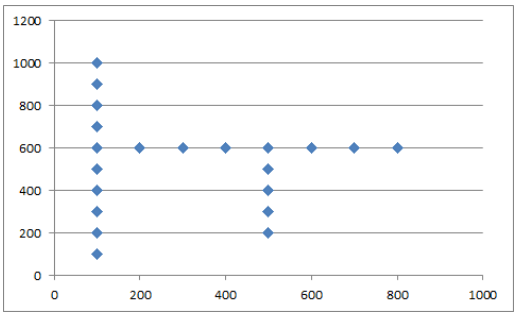


Figure 6.1: Points in the data

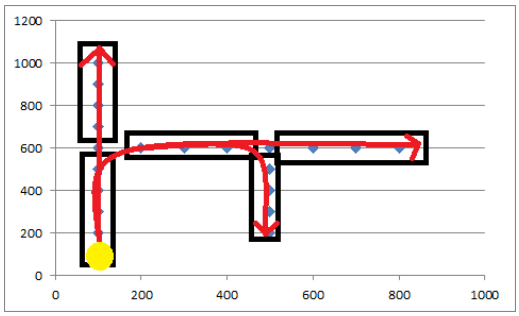


Figure 6.2: Logical clusters of these paths

We made this data set with the intention to test the method. It therefore has the different possible cases for a trajectory. The yellow dot is the beginning at it contains 3 different routes. Please note that the tracks in the graph are mirrored compared to the figures of the clusters below. This is because the y-axis is mirrored in the program. In the graph, the starting point is in the left under corner and in the figure of the clusters it is the left upper corner.

The basic data set consist of 740 data points that are all distributed over 3 different paths that all have 20 entries, so we have 60 paths in total. All the data points are 100 steps from each other. For the test we use the followings settings: step size of 50, variance of 20 and a grid size of 20 by 20. We we apply the method to this data (figure 6.3, which is equal to 6.1) we get exactly the clusters we expected (figure 6.4), five straight clusters.

To make the data a bit more challenging we randomised it a little. Every point in the data is randomised with a maximum of 20 steps in each direction. The data points will then look like the set in figure 6.5. We will see that also this set will lead to the 5 expected clusters. You can see in figure 6.6 that these lines are less straight than the previously created clusters. This is logical since the lines in this data set are also more random.

Finally we used a more dense and a more random data set as seen in figure 6.7. We also added another line to this set that also starts with the root but directly moves in another direction. For this test we made a data set containing 10 times more data points, with every data point approximately 10 steps from each other. Every point in this data is randomised with a maximum of 5 steps in each direction. When we use our method we see that it has some more difficulty with this set (figure 6.8). The clusters do not connect as nicely as they did before but it still results in the 6 clusters that we expected.

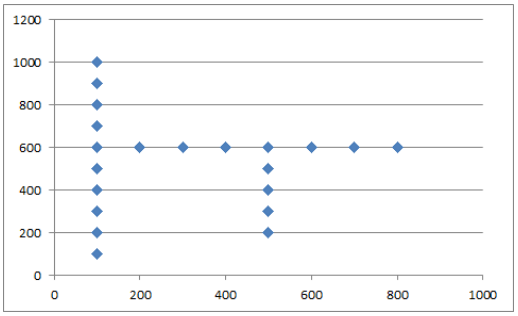


Figure 6.3: Points in the basic data

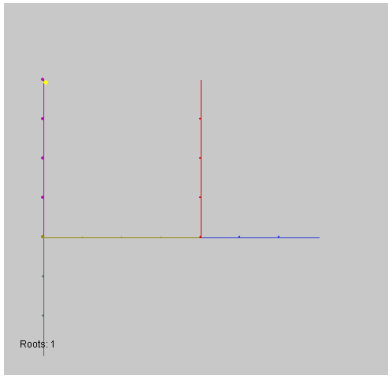


Figure 6.4: The clusters from the basic data

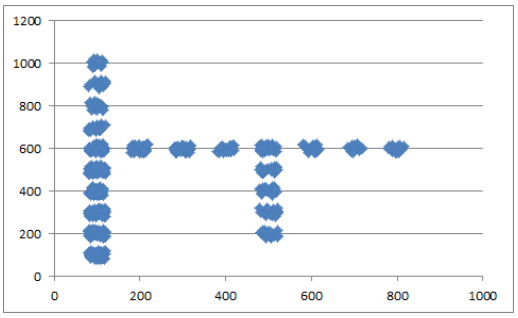


Figure 6.5: Points in the random data

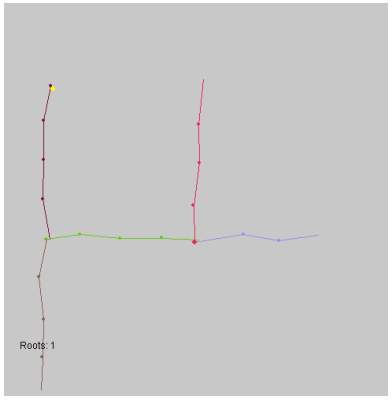


Figure 6.6: The clusters from the random data

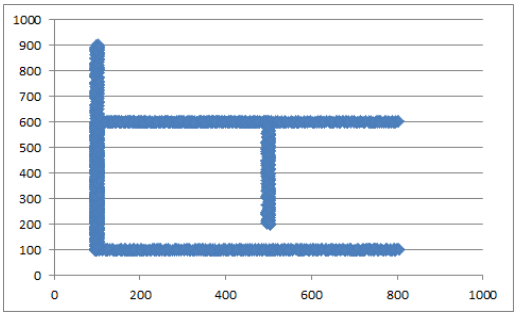


Figure 6.7: Points in the large data

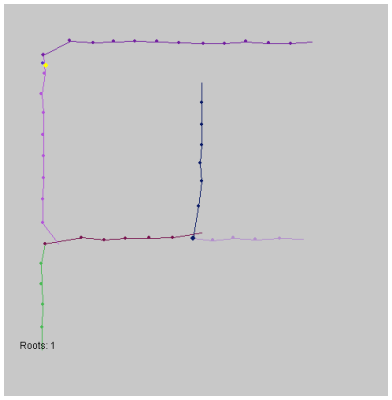


Figure 6.8: The clusters from the large random data

We can conclude that the method is working well for relatively small and simple data sets. We can see that with the most easy data set we retrieve a perfect set of clusters. When we used a random offspring to these points the method is still returning the clusters that we expected. Even in the larger set with an even greater randomness in the points, the method is still able to retrieve the correct clusters.

In the next section we will look at the method for a much larger and complicated environment.

6.2 Cologne

6.2.1 Introduction

We want to test our method in an environment that is more suitable to our problem definition. We want to show that the improvements and adjustments that we made had a positive influence on the outcome of the method. To do so we need a data set that is very large and has entries of objects in an urban environment. We will therefore use the traffic data of a large city.

6.2.2 Dataset

We will use the simulator named Sumo Sim, which is able to generate simulated traffic data and output this to an XML file. This means that it can generate a very large amount of data. There are a lot of data sets available and we decided to use the data set of Cologne for our tests. Cologne is a large city in Germany with a lot of traffic, this makes it a perfect dense urban environment for our tests. To give an indication of the density of the data, after around 1000 time steps, there are more than 2000 vehicles in the environment.

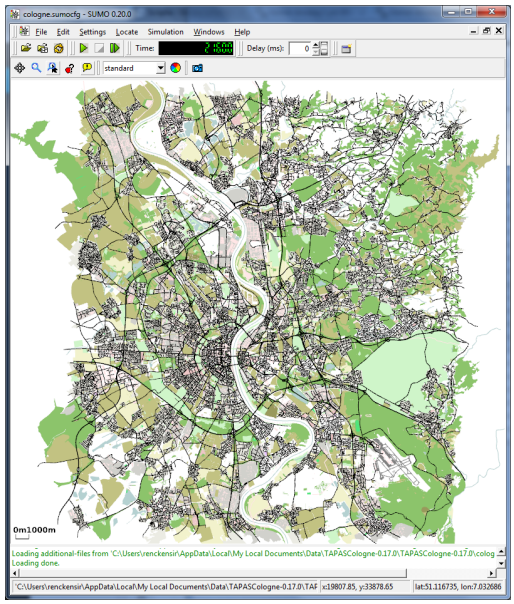


Figure 6.9: The Sumo Simulator

6.2.3 Experimental Setup

What we would like to know is the difference between our improved method and the original method proposed by Piciarelli et al. [14]. We will compare it to see if our proposed changes actually are an improvement to the original method. We will use the same data set for both methods, then we will determine whether it can detect anomalies better than the original method.

We created a training set of 3,000 objects. These objects are part of the Cologne data set, we will consider these objects as normal. We tested the method with 4 different test sets. The first set we used the complete training data as the test set for reference. The second set is a test data which has 30 normal objects, and these did not occur in the training set. The third and fourth set are the sets with 30 anomalous objects each. One of them has an anomalous speed and one has anomalous paths.

We will only test anomalous on the area of speed and tracks. We would also want to see whether the lifetime gives good results, but during the testing we realised that the use of the system time of the computer is not a good measure to capture the lifetime of an object. It gives different results on the same tests, for example when you run multiple tests simultaneously on one computer, the run time will be different from the same tests that were done consecutively. We will therefore not consider the lifetime in the experiments.

6.2.4 Inserted Anomalies

To ensure that we actually have some anomalies that can be detected, we will manually add these to the data. We will insert a total of 30 anomalous data entries to each anomalous data set:

- Anomalous tracks: 30 objects with a strange path. For this set we use exactly the test data, but we will flip the y-axis. Because we use Cologne, this results in a similar track set but is definitely not all on the exact roads in the environment. See also figure 6.10.

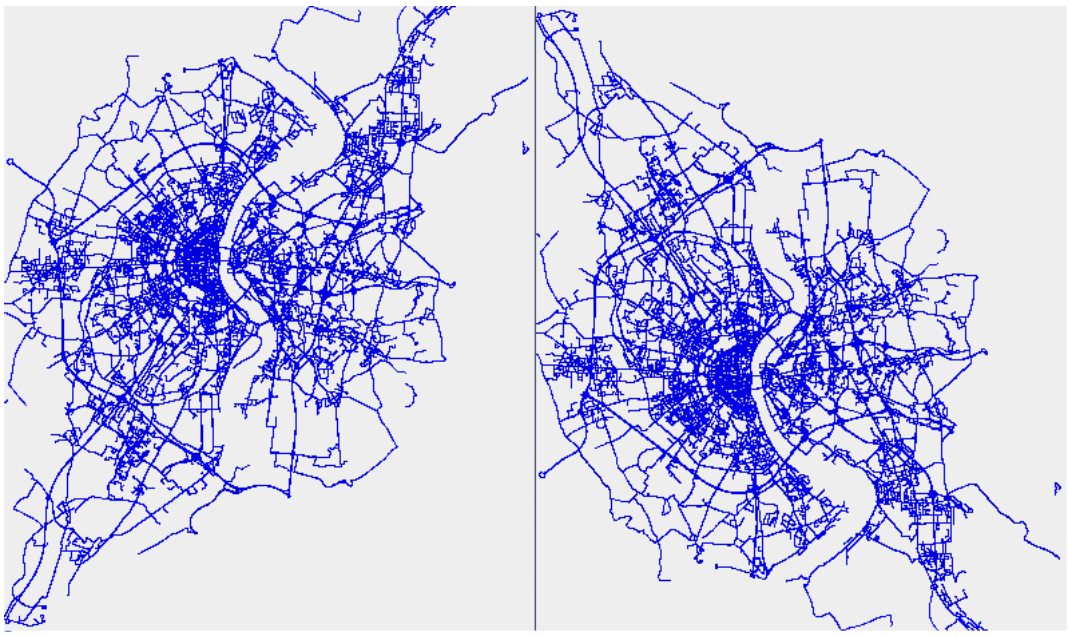


Figure 6.10: Left: Normal map of Cologne, Right: Anomalous map of Cologne

- Anomalous speed: 30 objects with strange speeds, a max speed of 5 instead of the normal max speed of 45. Unfortunately we could only set the speed to lower than normal since the simulator does not allow vehicles to drive faster than the speed limit.

6.2.5 Anomaly Score

As we discussed in section 4.8, we did not find an optimal anomaly score function. We will therefore use a basic function to determine the score.

For the score of the speed of objects we use the Euclidean distance function divided by the variance of the closest cluster. For the tracks we use the distance measure that calculates the distance from a track to an assigned cluster. When a track does not match a cluster anymore, the distance will become bigger over time since it moves away from the last assigned cluster.

6.2.6 Parameters

In order to test the data with our method we need to set a few parameters. It is difficult to set the best fitting parameters, moat of the values are guesses. For the variance of the tracks we chose a number that is larger than the step size. For the speed we set a very small number because these values are all very small since we use the distance distracted from the coordinates. For the update rate α we chose 5%. We parameter settings are the following:

- Step size: 2
- Variance tracks: 4
- Variance speed: 0,05
- Grid size: 20 x 20
- α: 0,05

6.2.7 Learning time

In table 6.1 we can see the time it took the different methods to learn of the model and test the specific test set. The adjusted method is much faster than the old method. This can be explained by the time that is saved by the step size constraint. Because of this constraint not all data needs to be processed. Using the training data as the test data takes the longest time since this data set is 100 times larger than the test sets. This process took very long for the old method and even quite a long time for the adjusted method. The test data and the anomalous tracks data (which are the same but flipped) took much shorter and are still much faster in the adjusted method. Also the anomalous speed data is much faster with the adjusted method than with old method. This data set takes somewhat longer than the previous two, this is because the objects drive at a slower speed and therefore stay longer in the environment and generate more coordinates.

Training	Testing	Original method	Adjusted method
Train data	Train data	> 24 hours	~4 hours
Train data	Test data	~1 hours	~20 minutes
Train data	Anomalous tracks data	~1 hour	~20 minutes
Train data	Anomalous speed data	~1,5 hours	~30 minutes

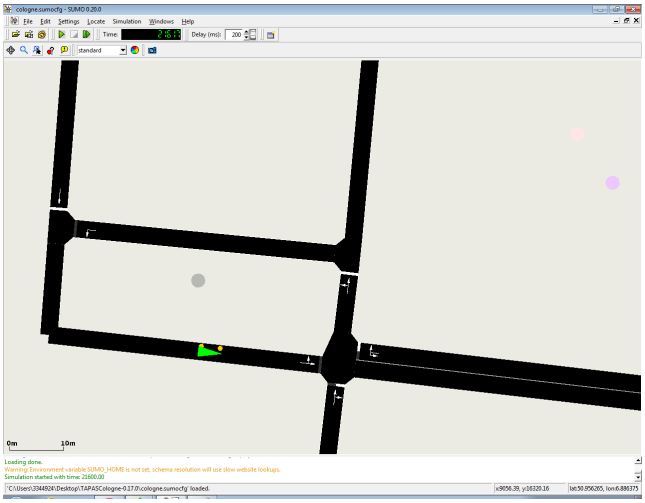
Table 6.1: Processing time methods

6.2.8 Limitations of the simulator

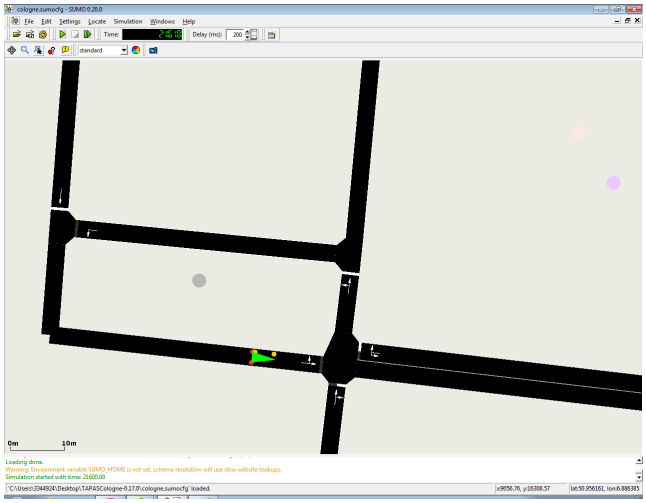
The simulator that we use is useful in many ways but it also comes with a few disadvantages and limitations. One of the disadvantages of this simulator is the way it handles crossroads. As it becomes clear in Figure 6.11, the simulator does not handle the crossroads as expected. When a vehicle is approaching the crossroads it moves at a normal speed, but when it crosses the street and makes a turn, it skips a part of the street. The simulator is built in a fashion that vehicles will never drive on the crossing but only jump over it to the street they are heading. This is visible between time step 3 and 4 in the figure. This results in sudden strange high values for the speed and also for the path that a vehicle is driving.

Another limitation of this simulator is the way it handles speed. The vehicles all keep to the speed limit in the environment unless their maximal speed is smaller. This means that we could only insert lower values for the speed as anomalous behaviour. We would like to run the tests on data with different kind of anomalous behaviour concerning the speed, but with this simulator we can only add smaller values to test on.

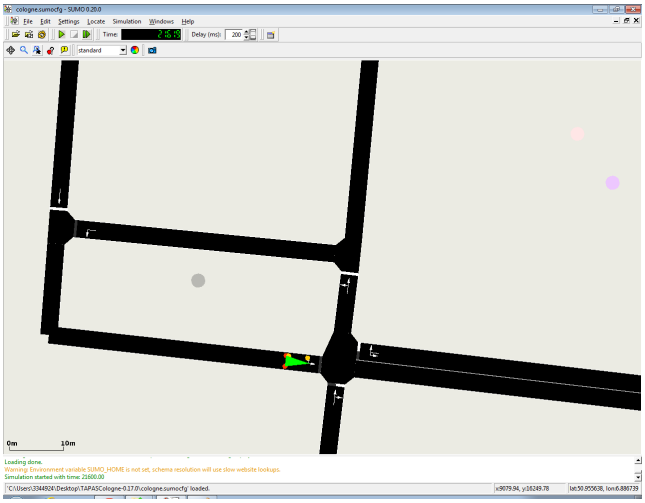
Another problem with the vehicles skipping over crossroads and ignoring speed limitations is the way the learning algorithm constructs paths when we use the simulator data. With our adjusted method, we use step step to determine when a point should be processed. When a vehicle jumps along the route, it is likely that the step after the jump is taken into account in the model. Since all vehicles drive at the same speed and all start at the beginning of a road where the jumps are, all of the routes of the objects are exactly the same. This means that when there is a vehicle with a lower speed on the road, it gets a high anomaly scores since it is not exactly the same as the other tracks.



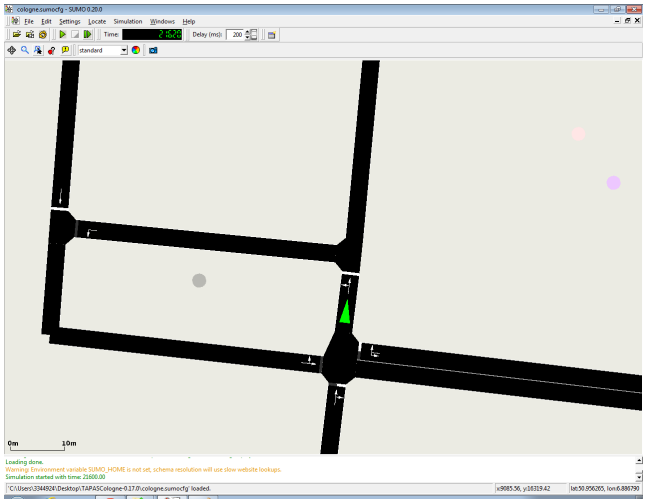
(a) timestep 1



(b) timestep 2



(c) timestep 3



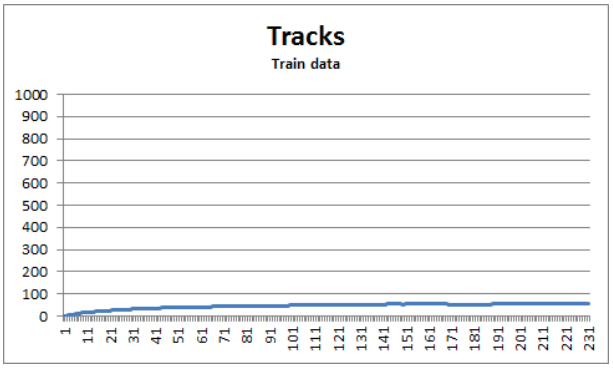
(d) timestep 4

Figure 6.11: Example of skipping a part of the route

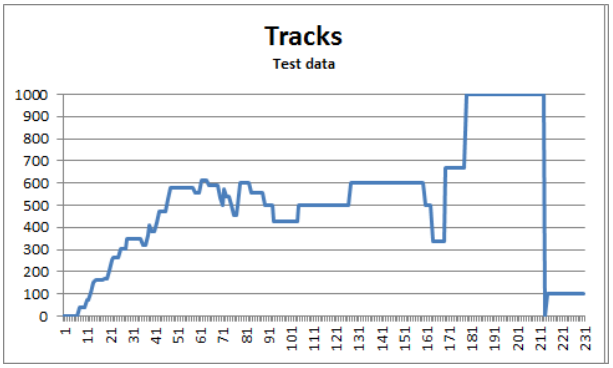
6.3 Results of the Adjusted Method

6.3.1 Results tests tracks graphs

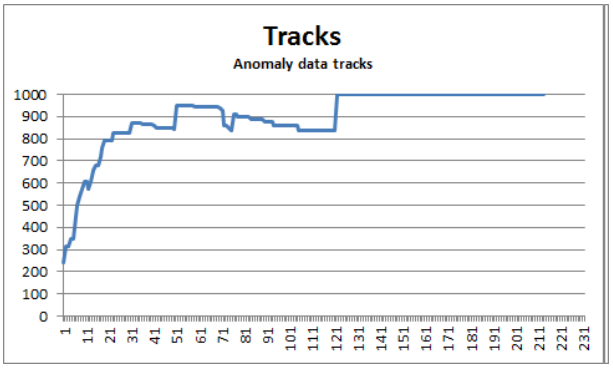
The graphs on this page shows the anomaly scores over time. At each time step the average score is shown for all the objects that are still in the environment. The x-axis represents the time and the y-axis the anomaly score. Therefore at the end of the graph the average is taken over fewer objects than at the beginning. This explains some jumps in the lines, especially near the end.



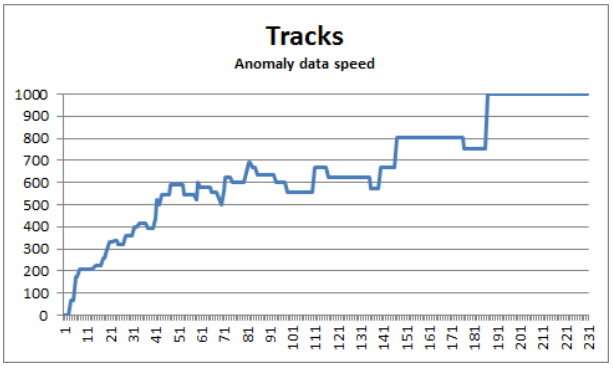
In this graph we used the same data for the training as for the testing. This resulted in a low overall score. This makes sense since it has been trained to learn those specific tracks. It is a good thing that the method recognises these tracks and assigns a low anomaly score.



In this graph we tested our trained model with a sample of 30 objects that were not present in the training data. The result of this is a line that has a small anomaly score at the beginning but it increases in time. We can conclude from this graph that the model needs more input to create a better model. It now only has a good model of the start of each track but misses data to create a good model of the tracks that are further away from the starting point.



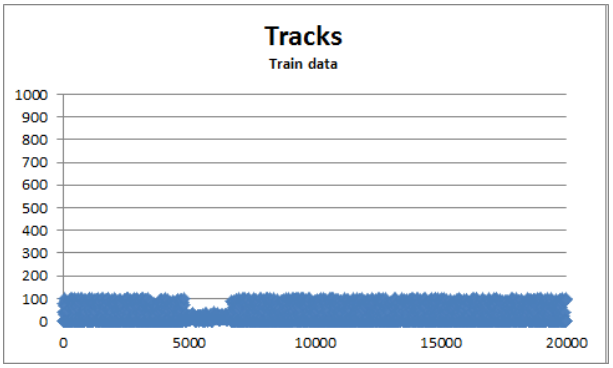
When we take a look at this graph we see the line retrieved from a set of anomalous paths. It is clear that in this graph the anomaly score is increasing much faster than with the normal data. After a while it will even no longer drop below the maximum score.



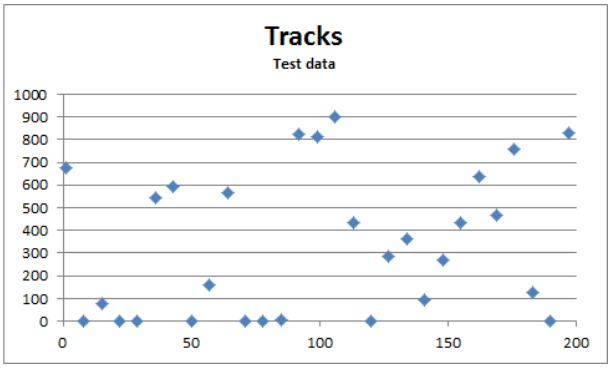
This is the graph of the test set with the anomalous speed. This graph is very similar to the graph of the test data and this is logical. It is exactly the same tracks but with another speed. Since it only registers a new point after a certain step size, this will result in the same graph as the normal paths.

6.3.2 Results tests tracks histograms

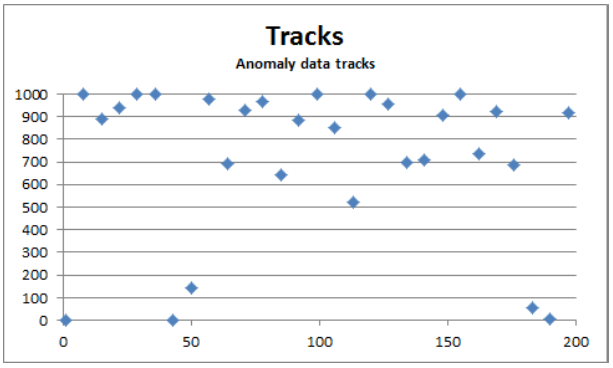
We can see more clearly the difference between the normal and anomalous data when we look at the histograms on this page. These histograms show the average anomaly score of a track. The x-axis represents the objects and the y-axis the average anomaly score.



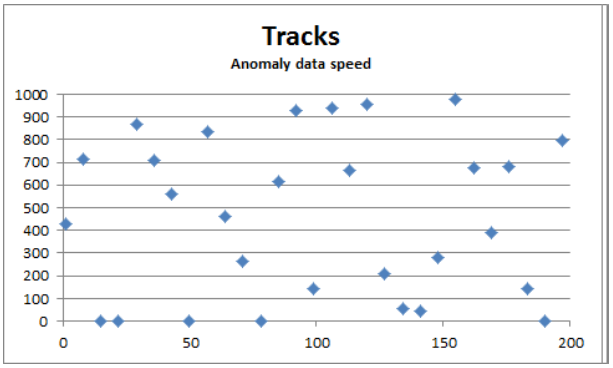
In the first histogram, we can see that all of the average anomaly scores are very low. This is again because the model is trained on these objects, therefore the anomaly score is low. This is a good result.



We we take a look at the histogram of the normal test data, we see that most of the scores are below half of the area. For this case holds, when we would have more training data we can possibly make a better model. Unfortunately we did not have the time to test this. With this better model the scores will be lower, more like the scores when we use the training data as the input for the testing.



When we use data with anomalous tracks, we retrieve this histogram. When we compare this histogram with the test data we can clearly see that the method gave the anomaly data higher scores. Most of these objects are at the upper half of the area. This means that the method successfully detected the anomalies.

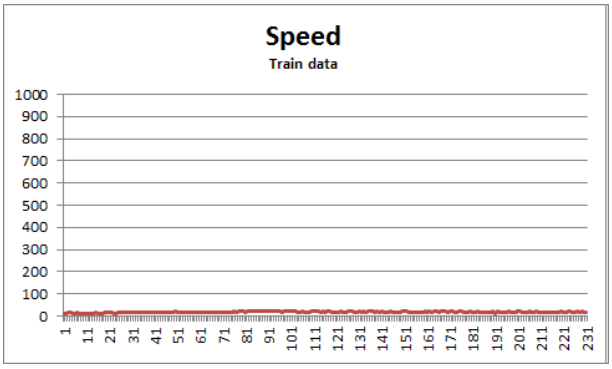


For the data with anomalous speed we get this histogram. This histogram is very similar to the histogram of the test data as expected. This has the same reason as mentioned above, the tracks are the same, only the speed has been varied.

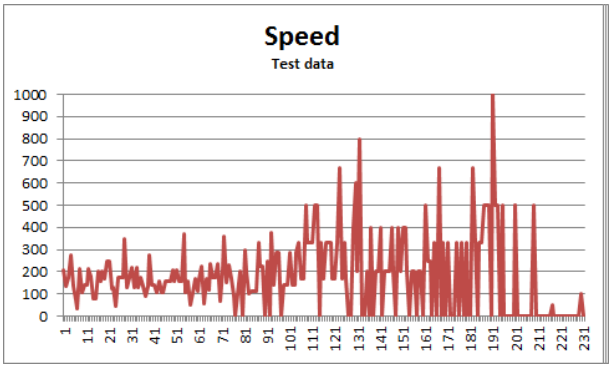
6.3.3 Results tests speed graphs

The x-axis represents the time and the y-axis the anomaly score.

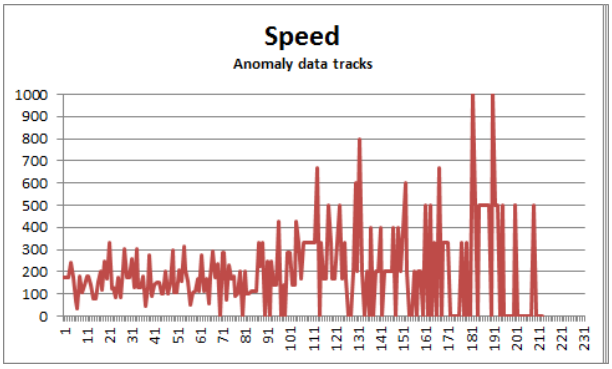
This is the graph acquired from the training data as input. All of the object have a low score. This is because the train data has learnt exactly the input for the test data, this is the logical result.



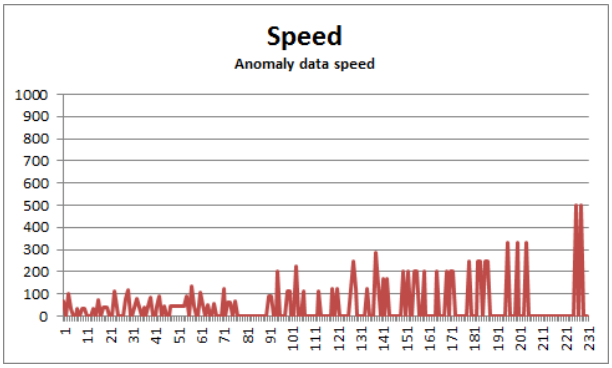
This graph shows the result of the normal test data as input. As is clear in the graph, the values are higher than in the previous graph. Towards the end of the graph the values become more extreme, this is because the average is taken over fewer objects. With more training data, this graph might have lower values.



This graph is almost exactly the same as the graph above. This is the graph of the anomalous tracks. The speed in these tracks are unchanged, therefore we get the same graph.



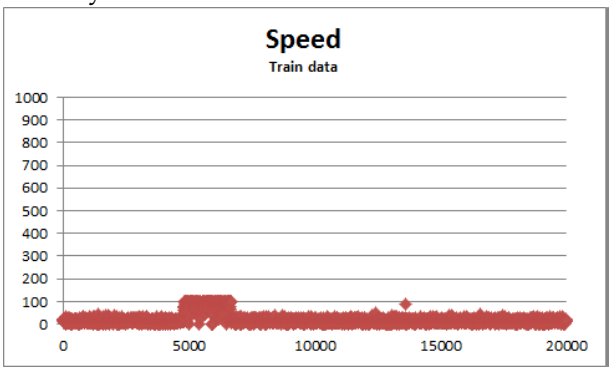
This graph shows the results of the anomalous speed data. At first sight it is strange to see that these values are lower than the values from the previous tests. It is explainable because the values of speed are already very small and with a smaller interval of change in the maximum speed, the values will fall more into the known model.



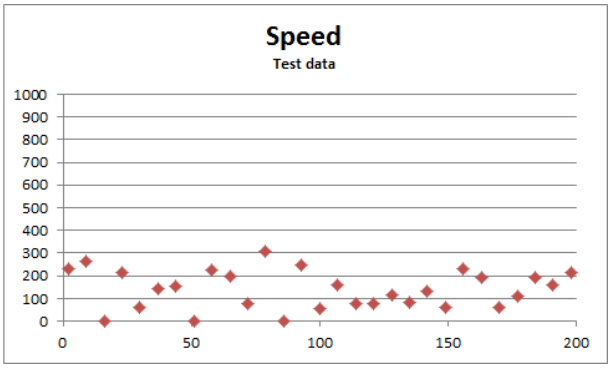
6.3.4 Results tests speed histograms

The x-axis represents the objects and the y-axis the average anomaly score.

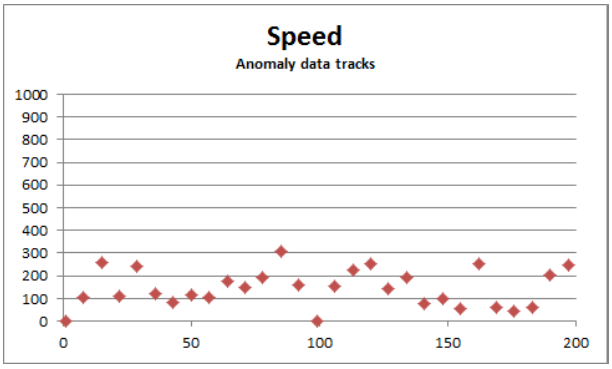
In this histogram of the training data as input we can see that all of the objects receive a small anomaly score as expected since the model was trained with this data.



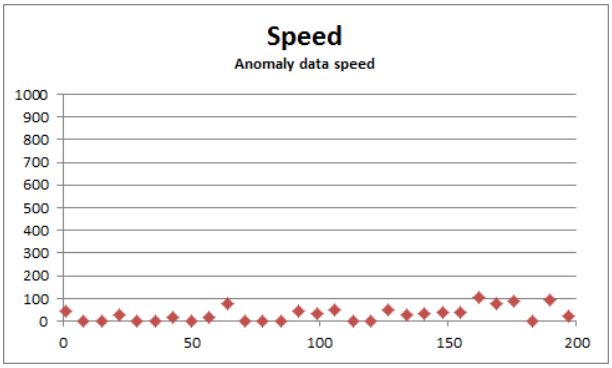
In this histogram, the averages are a bit higher than the previous. This is because the model had not enough input to create a perfect model. This is an approximation.



This histogram shows the results of the anomalous tracks. This gives the same results as the normal test data. This makes sense since the speed in these data sets are equal.



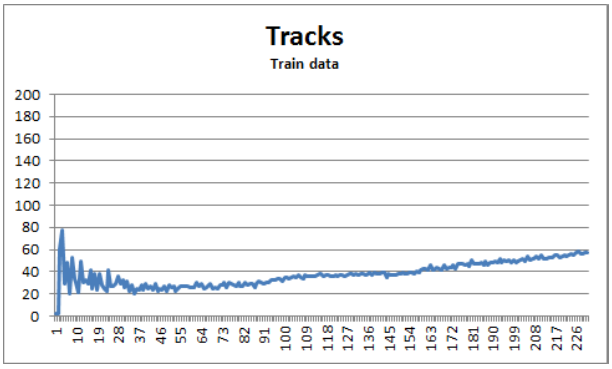
This last histogram shows the objects of the anomalous speed data set. We can see that the scores of these objects are lower than the scores for the previous two sets. This is caused by the limitation of simulator where we can only insert slower anomalous speeds. As a result, these values become smaller and have less fluctuations, this results in a lower anomaly scores.



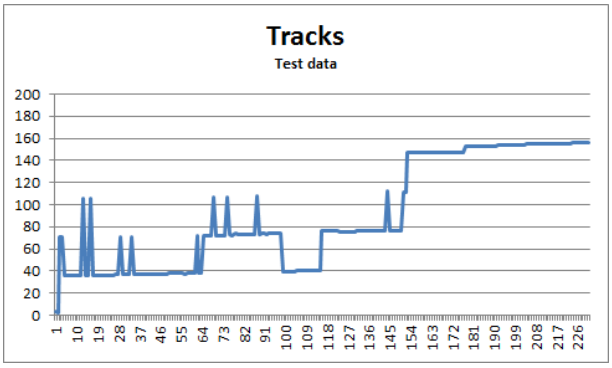
6.4 Results of the Original Method

6.4.1 Results tests tracks graphs

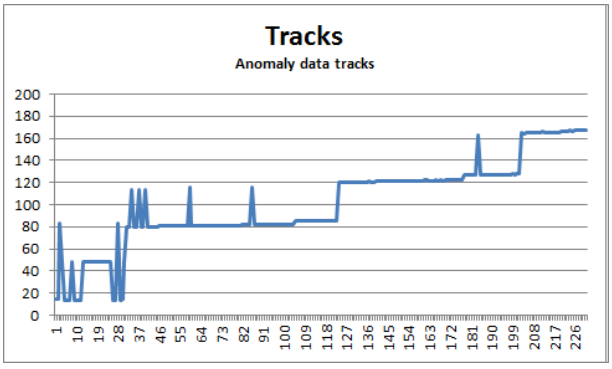
The x-axis represents the time and the y-axis the anomaly score.



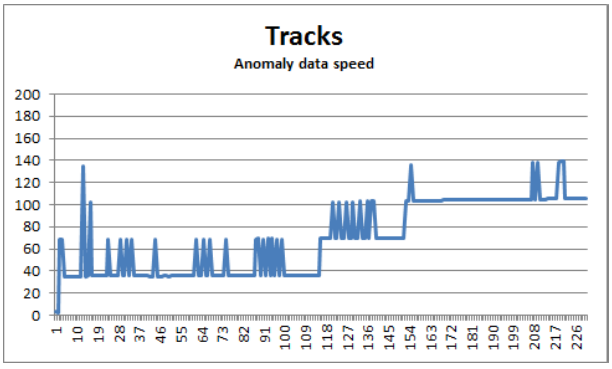
This graph shows the train data used as input. The scores are low in this graph. This is logical result, because the test data is the same as the train data.



When the test data is taken as input, the following graph is retrieved. The values in this graph are a bit more extreme which is explainable by the average taken over fewer objects at the end. Here also holds that it would give a better result if we have more training data.



This graph show the results of the anomalous tracks data. This graph is fairly similar to the previous graph. This means that there is little difference in the average over time between the normal and anomalous data.

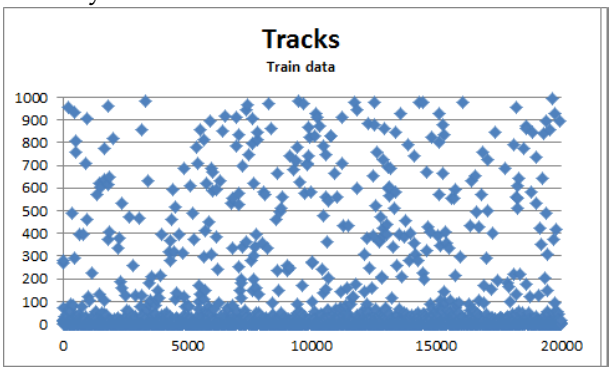


This graph is the result of the anomalous speed data. The results seem pretty similar to the results of the previous two graphs, just a few more low scores.

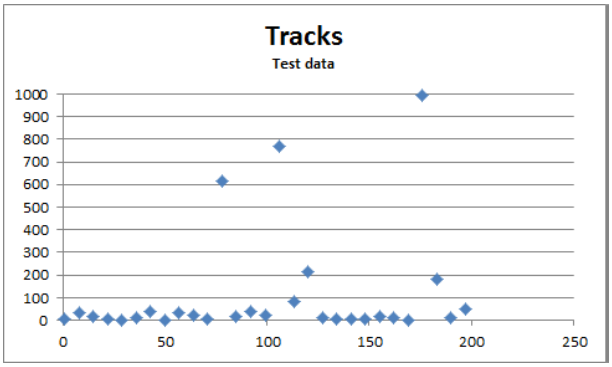
6.4.2 Results tests tracks histograms

The x-axis represents the objects and the y-axis the average anomaly score.

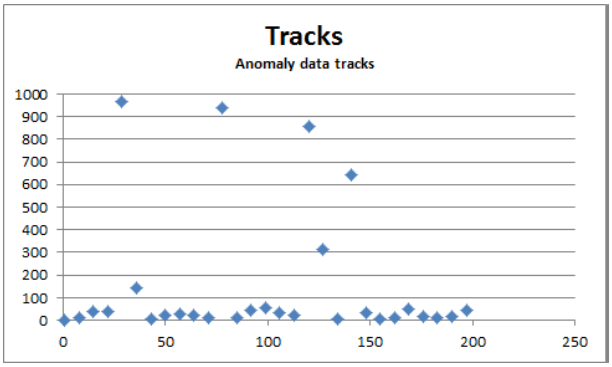
As is clear in this histogram, the scores of the different objects are very scattered in the graph. This means that even in the case that the test input is the same as the train input, the method will not recognise all of these objects as normal. Please note that a large amount of the data is in the lower part of this histogram even though this is not visually clear.



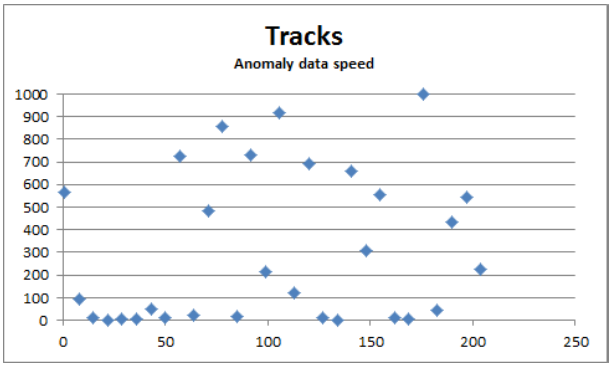
In this histogram the test data is shown. We can see that this gives a result where a lot of the objects are assigned a very low score. This is a good result, it seems to be a subset of the train data. This looks like the test test gives equally good results as the train data as the input.



This histogram shows the anomaly scores of the anomalous track data. It should give a large increase in the number of high anomaly scores compared to the previous histogram. This is not the case, it gives roughly the same scores as the normal data. This can be explained by the variation of the speed clusters that can become very high in this method. With a very high variation, all the objects can seem normal because they will always match with some track in this model.

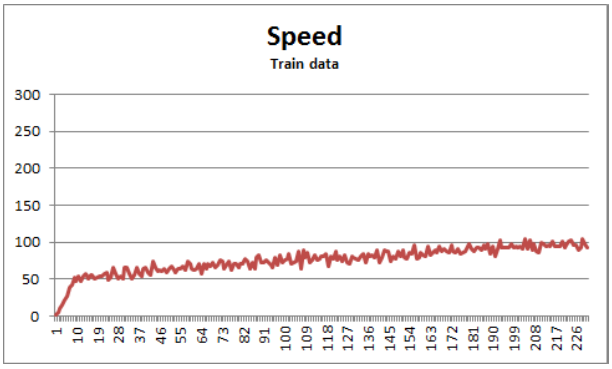


This histogram shows the anomalous speed data as input. This is a little counter intuitive that the results in this graph are a little more anomalous than the previous ones. This is due to the simulator. Since the tracks are learned in a certain manner as explained in the previous section, the tracks will differ from this pattern when they have a lower speed. This results in this histogram where there are more anomalies than with the normal test data.

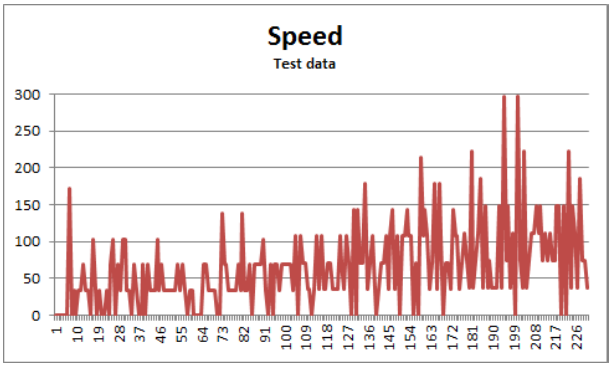


6.4.3 Results tests speed graphs

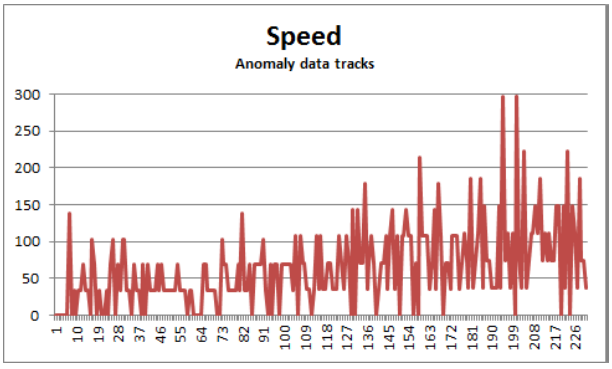
The x-axis represents the time and the y-axis the anomaly score.



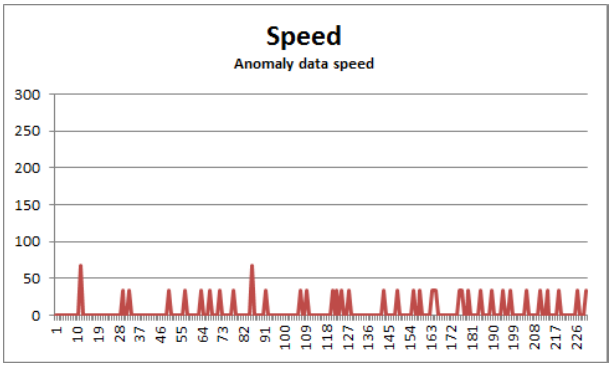
When we use the training data as the test input, the result for the speed is this graph. In this graph is shown that all the values will stay relatively low.



In this graph we see the normal test data as the input. This resulted in a line that is somewhat higher than the previous. It especially is more extreme, this is explained by the diminishing number of objects over which we calculate the average.



This graph is nearly the same as the previous one. This makes sense since the tracks are flipped but the speed is the same. We therefore receive the same results for the anomalous tracks data.

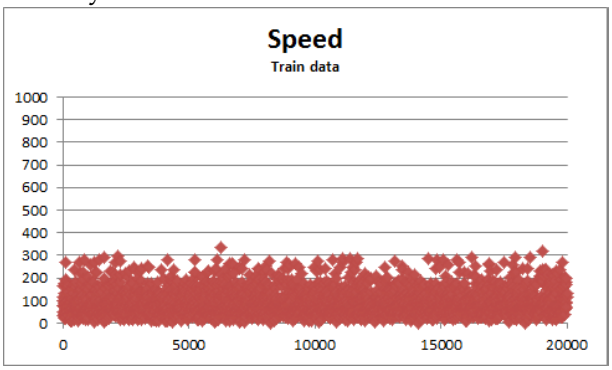


This is the result of the anomalous speed data. It has all very low scores, this has the same explanation as this occurred for the adjusted method. Since we deal with very small values for the speed, the anomalous lower values for the speed are actually better in the model since there are less extremes.

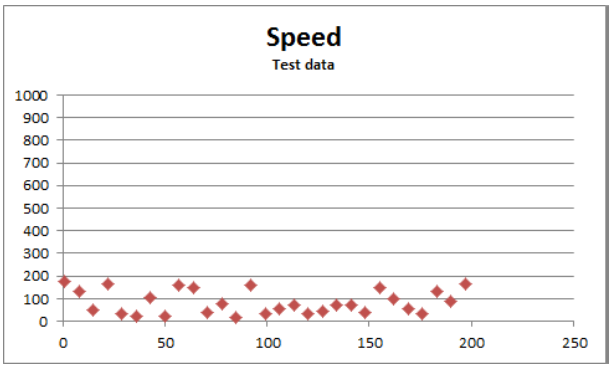
6.4.4 Results tests speed histograms

The x-axis represents the objects and the y-axis the average anomaly score.

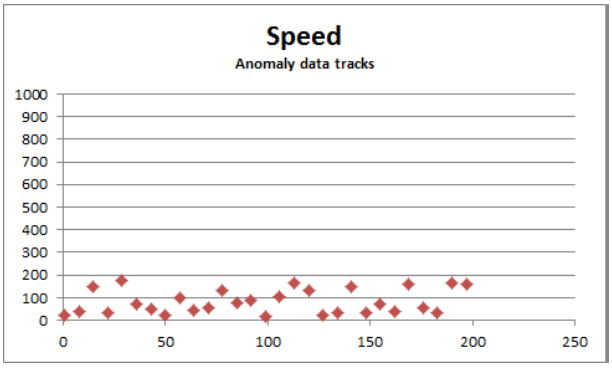
This histogram shows that all of the values retrieved from the training data as input are quite small. This is as expected from an input that has been trained on.



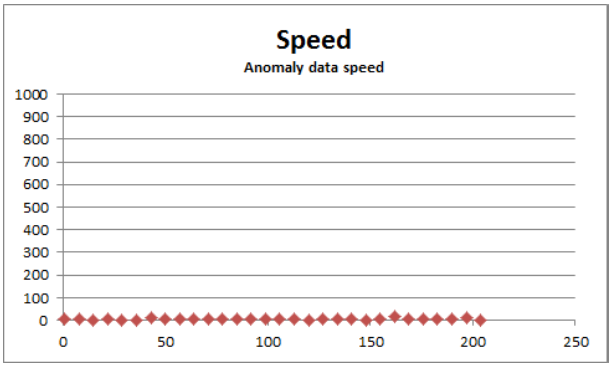
This histograms is the result of the normal test data. All values are very small and very similar to the histogram shown above, this is a good result.



The histogram of the anomalous track data shows approximately the same results as the previous test. The speed in these sets are equal, so this is an expected result.



These values are much smaller than the previous two tests. Since the values in the speed are all so small, the anomalous speed data gives even better results. In the optimal case, we want to have more extreme values in this histogram.



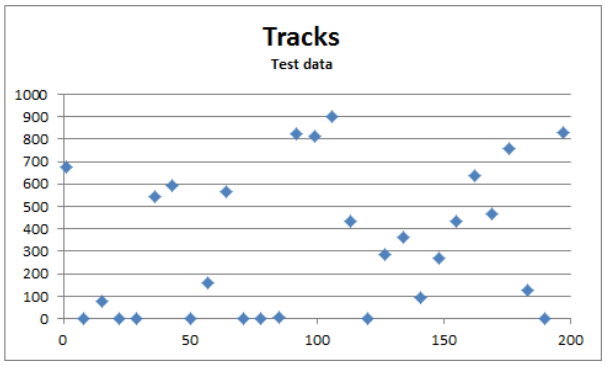
7 | Conclusion

7.1 Discussion of the Results

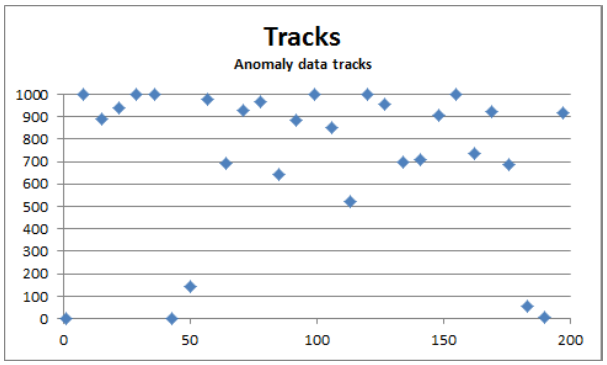
In order to see whether the adjustments that we made to the method are in fact improvements, we need to compare test results of the two methods. Please note that this comparison is not about the exact anomaly scores, but about the increase in scores between the normal and anomalous data.

7.1.1 Tracks

We start with comparing the results on the tracks part:

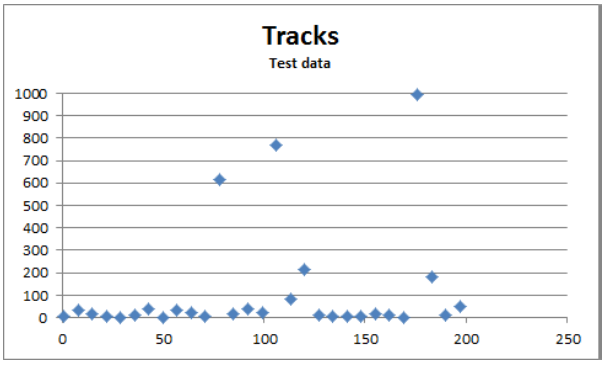


(a) Histogram of Normal Test Data

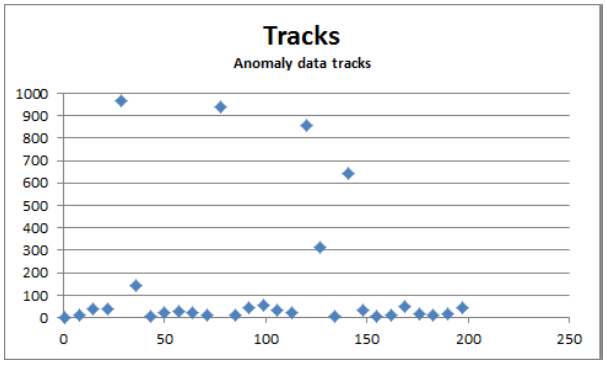


(b) Histogram of Anomalous Track Data

Figure 7.1: Histogram Tracks of the adjusted method



(a) Histogram of Normal Test Data



(b) Histogram of Anomalous Track Data

Figure 7.2: Histogram Tracks of the old method

We can see in the histograms in figure 7.1a and 7.1b of the adjusted method, there is a significant increase in the anomaly scores with the anomalous data. In the old method in figure 7.2a and 7.2b this

increase is not really present.

This is clearly visible when we run a students t-test on the sets. This t-test calculates the probability that two subsets are derived from the same set. In other words, the t-test value is a measure for the similarity between two sets. We use a two sided test with unequal variances since we are not dealing with a paired set. The test will pass when the two set are equal, with a p-value below 0.05. It will be rejected when the the p-value is above this value,

For the t-test on the two sets on the adjusted method, the p-value is 4.34247E-05. When we run the t-test on the old method, the p-value is 0.561269864.

These values tell us that the normal and anomalous data of the old method do not differ much from each other, the passed the norm of 0.05. The method is not effective in detecting anomalous data objects. The explanation for this is the variance handling in the old method. Since there is no limit for the variance, these values can become large. This means that a new data object will have a very high probability of being able to belong to a cluster, which will assign it to a non-anomalous status.

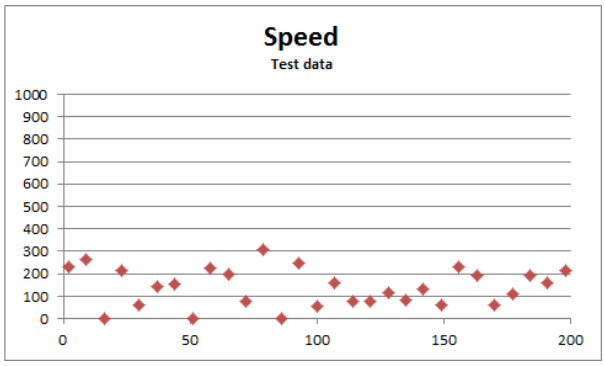
In the adjusted method, anomalous data objects can be better detected. The detection of anomalous data objects is improved through several adjustments on the algorithm, namely:

1. the starting grid that solved the problem of the too specific trees
2. the space dependence instead of the time dependence
3. the limit to the variation value of a cluster

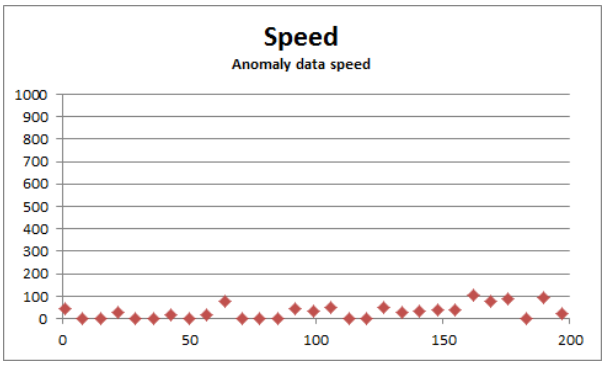
The p-value value for these sets is very low, which indicates no similarity between the two sets. The observed difference is very unlikely to occur if the means were in fact equal. Since the values are changed as expected, more anomalies, we can conclude that the method is improved on the part of the tracks.

7.1.2 Speed

For the comparison of the speed:



(a) Histogram of Normal Test Data

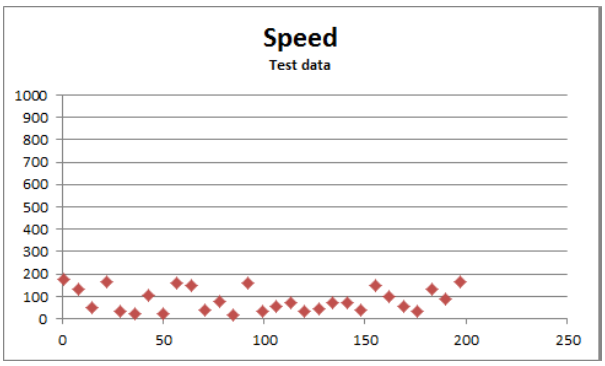


(b) Histogram of Anomalous Speed Data

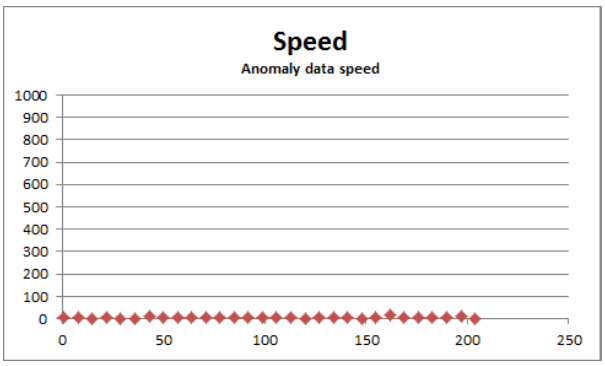
Figure 7.3: Histogram Speed of the adjusted method

When we take a look at these histograms, it is clear that they do not differ much from each other. In both methods the test data has small values as the average anomaly scores. But the histograms with the anomalous speed data shows even smaller anomaly scores. As explained in the previous chapter, this is a result of the simulator. Since it is only possible to enter smaller values for the speed, it is impossible to add speed that is unusually high. Without this added to the data, it consists of only small values. All of the values are already small and the smaller speed will lead to less extreme values that result in overall small anomaly scores.

When we run a t-test on these experiments we find the p-values 2.45547E-07 and 9.76736E-09 for respectively the adjusted method and the old method. Both methods are based on the same technique,



(a) Histogram of Normal Test Data



(b) Histogram of Anomalous Speed Data

Figure 7.4: Histogram Speed of the old method

the main difference is the boundary of the variation of objects. In this case where we can only add lower values of the speed, there is not much difference between the methods. In both cases there is a significant difference between the normal data and the anomalous data, but both are not working as desired.

7.2 Summary

We will summarise this thesis by looking back at the sub problems formulated in section 1.3.

We started this by looking at all the different methods that are available for anomaly detection. We considered the work of others in the field of the use of anomaly detection on security applications. We also looked into the different kind of methods for anomaly detection in general.

Afterwards we made a list of the requirements of the method. We considered that the method should be able to work on a UaV, that it only received coordinates as input and that the processing time should be very fast. The model that was created is a typical two phase model. First the learning phase and afterwards the testing phase. This testing phase in particular needs to be fast, this should be used real time. The different features that should be learned by the model were determined, it will capture tracks, speed and lifetime in the environment.

After the requirements of the method were determined, we chose a method that we would adjust for this specific case. From all the different anomaly detection techniques that are available, the cluster algorithms seems to be the most promising. The choice for these algorithms is based on the fast detection phase, the small amount of memory needed and the ability to train a model without labelled data. We started an implementation by taking the clustering method of Piciarelli. We applied this clustering method to the three features that we capture for the model. We made a number of adjustments to the model to improve the functioning on our specific environments. We cleared some ambiguities from the paper, such as the addition of points to a cluster and the exact formula for the distance measure. We also added some solutions to the method of Piciarelli such as the solution to the problem that there is a lot of data needed to create a model with enough information. Therefore we set a grid than an object need to cross in order to be taken into account for the model. This leads to fewer starting points, since they are all on the border of the grids. We also added a part to the algorithm that solves the time dependence problem of the original method. We implemented a step size that an object need to cross before its coordinates are processed by the method for the model. The last major adjustment is the variation control. In the original method, the variance of the clusters is unlimited. This leads to illogical models and strange tracks. We therefore use a limit for the variance of the clusters that they can not exceed.

To test the improvements that we added to the method we performed a series of tests. The outcome of these tests showed that our adjusted method is an improvement on the original method. Mainly of the part of the tracks there is a clear improvement on the detection of anomalous objects. When we also

take into account the processing time of the two methods, our new adjusted method is much faster and it gives better results. Even though some serious improvements have been made, the method is still not completely working as desired. It can detect anomalies better than in the original method, but still the results itself are not perfect.

7.3 Future Work

As discussed in the summary, the method is working better than it was in the original method but there is still a lot to do to make this method work for real life purposes. We will discuss a few possible fields of improvements for future work.

The method now is based on very simple behaviour. We only take into account individual behaviour. If it would be possible considering the process time, the memory and the data available it would be an interesting addition to see whether individual behaviour can lead to better results. Possible fields of interest could be to model groups moving together and those who wander off, the number of people in an area, behaviour of an object compared to the rest of the objects in the neighbourhood regardless of the learnt model.

Another adjustment might be the used model in general, we used a very simple method to process the data. It is only based on individual coordinates. The simple method is necessary considering the requirements, but if it could be possible, a more complex method might be better in the learning and detecting of behaviour. A suggestions would be to use a Hidden Markov Model to learn the paths in the environment. The question remains whether this would be feasible since it is dealing with a very large amount of data.

A problem that we still encountered was the need of more data to create a good model. One solution is to give more time and input to the method to learn the model. This would lead to a better model of the environment and therefore also to better results. Another option is to change the way the data is used in the method. We already only use the data when an object crosses a grid line to assign the corresponding starting tree from there. Another option might be to assign a new tree every time an object crosses a grid border. This means a loss of information about the tree it was assigned to in the previous grid, but it increases the amount of data for each tree very much. It also removes the problem of the paths becoming too specific once it is further down the paths in the assigned tree.

For the speed and the time that we use in this methods we use the information that we can extract from the coordinates. This information is usable but it would be much more reliable if we use a data set that already has these values available. Tests with this addition can show if these values can be used better than the extracted information. Also testing with a better anomalous data set of the speed would give some insight in the functioning of the method.

Bibliography

[1] Alon, J., Sclaroff, S., Kollios, G., & Pavlovic, V. (2003, June). Discovering clusters in motion time-series data. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (Vol. 1, pp. I-375). IEEE.

[2] Brax, C., Niklasson, L., & Smedberg, M. (2008, June). Finding behavioural anomalies in public areas using video surveillance data. In *Information Fusion, 2008 11th International Conference on* (pp. 1-8). IEEE.

[3] Burghouts, G. J., & Marck, J. W. (2011). Reasoning about threats: From observables to situation assessment. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(5), 608-616.

[4] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR), 41(3)*, 15.

[5] Chandola, V., Banerjee, A., & Kumar, V. (2012). Anomaly detection for discrete sequences: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 24(5), 823-839.

[6] Francois, J. M., Leduc, G., & Martin, S. (2004). Learning movement patterns in mobile networks: a generic method. European Wireless 2004.

[7] Fu, Z., Hu, W., & Tan, T. (2005, September). Similarity based vehicle trajectory clustering and anomaly detection. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on* (Vol. 2, pp. II-602). IEEE.

[8] Guralnik, V., & Haigh, K. Z. (2002, July). Learning models of human behaviour with sequential patterns. In *Proceedings of the AAAI-02 workshop ï£¡Automation as Caregiver* (pp. 24-30).

[9] Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., & Maybank, S. (2006). A system for learning statistical motion patterns. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 28(9), 1450-1464.

[10] Johnson, N., & Hogg, D. (1996). Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8), 609-615.

[11] Makris, D., & Ellis, T. (2005). Learning semantic scene models from observing activity in visual surveillance. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(3), 397-408.

[12] Mori, T., Fujii, A., Shimosaka, M., Noguchi, H., & Sato, T. (2007, December). Typical behavior patterns extraction and anomaly detection algorithm based on accumulated home sensor data. In *Future Generation Communication and Networking (FGCN 2007)* (Vol. 2, pp. 12-18). IEEE.

[13] Morris, B. T., & Trivedi, M. M. (2008). A survey of vision-based trajectory learning and analysis for surveillance. Circuits and Systems for Video Technology, IEEE Transactions on, 18(8), 1114-1127.

[14] Piciarelli, C., & Foresti, G. L. (2006). On-line trajectory clustering for anomalous events detection. *Pattern Recognition Letters*, 27(15), 1835-1842.

[15] Piciarelli, C., Micheloni, C., & Foresti, G. L. (2008). Trajectory-based anomalous event detection. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11), 1544-1554.

[16] Porikli, F., & Haga, T. (2004, June). Event detection by eigenvector decomposition using object and frame features. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on* (pp. 114-114). IEEE.

[17] Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. *ASSP Magazine, IEEE, 3*(1), 4-16.

[18] Song, X., Wu, M., Jermaine, C., & Ranka, S. (2007). Conditional anomaly detection. *Knowledge and Data Engineering, IEEE Transactions on*, 19(5), 631-645.

[19] Swears, E., Hoogs, A., & Perera, A. A. (2008, January). Learning motion patterns in surveillance video using hmm clustering. In *Motion and video Computing, 2008. WMVC 2008. IEEE Workshop on* (pp. 1-8). IEEE.

[20] Teng, H. S., Chen, K., & Lu, S. C. (1990, May). Adaptive real-time anomaly detection using inductively generated sequential patterns. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on* (pp. 278-284). IEEE.

[21] Xiang, T., & Gong, S. (2005, October). Video behaviour profiling and abnormality detection without manual labelling. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on* (Vol. 2, pp. 1238-1245). IEEE.

[22] Zhang, Z., Huang, K., & Tan, T. (2006, August). Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on* (Vol. 3, pp. 1135-1138). IEEE.