TECHNISCHE UNIVERSITÄT
ILMENAU
Department of Computer Science and Automation
Systems and Software Engineering Group

Master Thesis

Identification of Trajectory Anomalies in Uncertain Spatiotemporal Data

Registration Date: 1 April 2020
Submission Date: XX August 2020

Supervisor: Prof. Dr.-Ing. Kai-Uwe Sattler (TU Ilmenau)
Supervisor: Prof. Igor Anikin (KNRTU-KAI)

Submitted by: Mardanova Aigul
Matriculation Number 62106
aigul.mardanova@tu-ilmenau.de

Abstract

Abstract content

Contents

Abbreviations

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Nowadays spatiotemporal (ST) data analytics plays an important role in different applications, based on Geographic Information Systems (GIS). Recent advances in GIS and, in particular, in GIS technologies and infrastructure have made cities smarter. And Intellectual Transport Systems (ITS) with urban traffic analysis are one of the most attractive applications in a smart city [3]. Intelligence surveillance in smart cities has rapidly progressed in last decade [4]. More and more roads and public areas are getting equipped with monitoring video cameras, amount of publicly available video data increases further [5]. Automatic analysis in Traffic Video Surveillance (TVS) receives increasingly more attention [6].

Nowadays there are many tasks and applications of urban traffic analysis and, according to [4], tracking vehicles behavior using image processing of videos is one of the promising approaches. One of the main research approaches in urban traffic analysis, which works with data from monitoring video cameras, is mining frequent trajectory patterns from the ST data representing a traffic flow, because extracted trajectories can be afterwards applied to automatic visual surveillance, traffic management, suspicious activity detection, etc. [7]. Another important sub-category of traffic analysis, which has become a commended task in many applications in smart cities, is an identification of trajectory anomalies [4]. Anomaly is traditionally described as a data instance that remarkably deviates from the majority of data instances in a data set [8]. In TVS domain an anomalous activity refers to events violating the common rules [6]. Such unusual

traffic patterns, which do not conform to expected behavior, reflect abnormal traffic streams on road networks and thus provide useful, important and valuable information [4]. For instance, when a traffic incident or jam happens, traffic flow changes suddenly, and this will be reflected by deviations from the normative activity patterns. That means that recognizing outliers can be useful in detecting traffic incidents. However, in the context of huge amounts of data to be processed, or information overload in other words, manual solutions are infeasible nowadays due to high complexity and high time consumption, and researchers look for automatic or semi-automatic intelligent methodologies to solve these tasks to minimize the required involvement of the human operator [9].

As stated in recent researches in the field of traffic data analysis, it is significant in many applications, including ITS, to take into consideration uncertainty of data. The reasons of data uncertainty can be imprecisions in measurements and inexactitude of observations. In case of acquiring trajectory data from video enforcement cameras data uncertainty can be caused by limitations of used devices [10].

1.1 Problem Statement

As it was mentioned above, ST data analytics plays an important role in everyday life, and the process of extracting useful information from ST data is one of the most significant challenges in traffic data mining. Since ST trajectory data is multi-dimensional and spatiotemporally related, traditional data mining approaches, proposed for static, single and independent data, are inefficient and inappropriate in that case [11].

The main purpose of the work in this thesis is to implement a framework for frequent trajectory patterns mining and identification of trajectory outliers in a three dimensional ST trajectory data, extracted from video surveillance cameras. A video from surveillance cameras will be processed in a tracking system, which extracts vehicle trajectories and converts them into vectors containing tracking points. The implemented method needs to be evaluated in terms of accuracy, performance, and suggest an improvement to increase the accuracy of results in context input data particularities.

In order to achieve the main objectives, following sub-tasks need to be performed:

- Perform state-of-the-art review of existing approaches and choose a method to implement frequent trajectories extraction and anomalies detection;
- Investigate and suggest an improvement of the chosen algorithm to increase accuracy of results for data from video surveillance cameras;
- Implement a framework with the selected algorithm;
- Perform evaluation of implemented algorithm in terms of performance and accuracy.

In this thesis, we will focus on following types of anomalies:

- Anomalous trajectories with anomalous spatial information. This category covers trajectories with abnormal spatial behavior, such as illegal U-turns on the intersection, double solid line crossing, driving in an opposite direction.
- Anomalous trajectories with anomalous spatiotemporal information. This type corresponds to situations where the spatial information can be considered as a normal, but adding a temporal information converts the trajectory into an abnormal one, for example: moving with an anomalously high or low speed, unexpected, emergency stops.

1.2 Contribution

The main contribution of this work is ...

include or not?

1.3 Thesis Structure

The rest of this thesis work is structured as follows. The whole paper is organized into 6 parts. Chapter 2 introduces the background and terminologies

used in the thesis work. Chapter 3 performs the State-of-the-art analysis of existing approaches. Chapter 4 presents the concept of an implemented framework, describes input data structure and input data processing and provides the detailed description of the implementation part. Chapter 5 presents experimental results and evaluation of implemented approach. Chapter 6 gives conclusion and discussions on possible further perspectives.

Chapter 2

Basic Knowledge

2.1 Input Data Sources

Tasks of frequent trajectories identification and outliers detection can be applied to different data sources, for example: GPS (Global Positioning System) devices and sensor networks, then trajectory data is collected by sensors on moving objects, which periodically transmit information about location over time, or video traffic surveillance cameras. This work will focus on working with latter type of input data sources.

Video data from enforcement cameras is considered as a raw data and is not used directly as an input for implemented system. Raw video processing is done in a stand-alone tracking system. Tracking system takes raw video from enforcement cameras and handles it to perform the objects detection and converting the trajectory into a number of tracking points on images. Tracking points, containing such information as vehicle ID, timestamp, spatial coordinates, are used as an input.

2.2 Trajectory Definition

Trajectories can be described as multi-dimensional sequences containing a temporally ordered list of locations along with any additional information [8]. So, since a trajectory, denoted as τ, represents consecutive positions of a moving target

object in temporal domain, in a case of a single-camera surveillance data, it can be defined as:

$$\tau = (x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n), \tag{2.2.1}$$

where (x_i, y_i) denotes the position of the target object in the image at time t_i [7]. According to this, trajectories can be represented as a sequence of 3D points, where 2D object is for geometric coordinates and the third dimension stores the time [12].

Generally, trajectory data is raw and contain only minimum information such as position and time as well as the identifier of the tracking object. This information can be easily augmented by such detailed information as speed, acceleration and direction, since they can be extracted from the initial trajectory data [13].

2.3 Trajectory Anomaly Definition

Twenty-four-hour recording video surveillance cameras produce massive amounts of data about moving objects, and that increases the possibility that along with the normally behaving objects some of the moving objects will demonstrate abnormal behavior. Such exceptional behaviors can also be named as outliers, anomalies, abnormalities, exceptions, novelties or deviants [14][15]. Notwithstanding that no standardized way of deviation characterization exists, in statistics following definition can be found [16]:

> "An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs".

Trajectory anomalies can be described as traffic flow patterns, which significantly deviate from some normal behavior pattern or, in other words, inconsistent with the rest of traffic behavioral patterns.

The process of outlier detection is intended to reveal unusual patterns that drastically differ from majority of samples in order to process them further in an appropriate way [14]. Also anomalous to normality activity patterns ratio

should be relatively small in order to be able to distinguish abnormalities from the dominating normal patterns.

2.3.1 Trajectory Anomalies Classification

According to the literature, trajectory anomalies can be categorized as follows [15][17]:

- *Point anomaly* - represents the simplest type of anomalies. Corresponds to an individual data instance which is considered as an abnormal one with respect to the rest of data, since it deviates significantly from all other samples in a data set. For example, a non-moving car on a busy road.
- *Contextual anomaly* - corresponds to a data instance which is considered as anomalous in a specific context, but as normal otherwise. It can be also understood as a point anomaly in a neighboring of the data point itself. Contextual anomalies are also referred as conditional anomalies and represent the most common group of categories applicable to spatial and time-series data. For example, trajectories can be classified based on the spatial data (coordinates) in the scope of a time. Examples of a contextual anomaly may be trajectories of a vehicle moving with a much higher speed comparing to others in the same traffic flow or of a vehicle driving in the opposite direction.
- *Collective anomalies* - a set of data instances, cooccurence of which as a group is considered as an anomaly with respect to the whole data set, while each data instance individually does not necessarily represent an anomaly. The given definition can be simplified to a set of neighboring point anomalies or context anomalies. Collective anomalies can only be applied to data sets with a relation between data instances.

The other way of trajectory outliers systematization may be dividing them into following categories according to the properties, which were used to perform classification:

- *Spatial trajectory anomaly* - classification process takes into consideration only spatial information of moving object trajectories, such as position coordinates. Examples of spatial anomalies can be illegal U-turns, double line crossing or moving in an opposite direction.

- *Temporal trajectory anomaly* - corresponds to anomalies detected by analyzing only temporal characteristics of trajectories, such as duration, time of moving. For example, a trajectory with significantly long duration or a trajectory appearing at an anomalous time.

- *Spatiotemporal trajectory anomaly* - can be detected by analyzing spatial and temporal information in aggregate. Examples of ST anomalies can be vehicles moving with a considerably high speed comparing with majority of trajectories. Also such anomalies can be detected in the case of a contra-flow traffic systems with a reversing traffic light anomalous trajectories: since for such a line allowed direction changes according to some known or learned schedule, classifier can analyze the trajectory direction together with a temporal information.

In accordance with the second classification, the current work will focus on determining trajectory anomalies of the first and third types (spatial and spatiotemporal trajectory anomalies).

2.4 Challenges

Since ST data differs from other types of data in many aspects, challenges are related to the used data type. A unique quality of it is that ST data instances are not independent and identically distributed, as it is usually assumed to be in many of existing data mining approaches. On the contrary, ST data instances, related to observations made by nearby locations and time, are structurally correlated with each other in context of space and time, and it is important to take into consideration the presence of dependencies among measurements in these dimensions. Consequently, many of the existing data mining approaches are not applicable to ST data, since ignoring the aforementioned characteristics can result

in poor accuracy of results. This leads to the necessity of investigating and using different methods for processing such a data to preserve all the relations between information domains [8].

It should be noted that the chosen type of an input source leads to difficulties in processing. Since trajectory data is acquired by video enforcement cameras, the first problem is an uncertainty of a location as a result of limitations in measurement accuracy of the used cameras, resolution and the quality of the received images or frame jitter [5]. Moreover, enforcement cameras are placed at some fixed locations on an intersection, because of that one of the particularities of used data are pose and perspective, which can cause challenges while dealing with input video data [17]. The view angle of the camera in respect to the scene ground plane and distance between the tracked object and camera can affect the performance by decreasing the accuracy of objects detection and tracking: the smaller the angle - the bigger is the problem of determining the center of an object [4][5]. Tracked objects can drive in and out of the camera view, but be still detected while partially visible. This can cause trajectory changes on the borders of the camera scene: displacing and shifting of vehicle trajectories depending on the location of an object in respect to the camera [5]. Quality of the performed trajectory analysis also depends on the input trajectory data, including: quality of used cameras, quality of a tracking system, which converts a video data into a list of trajectories consisting of tracking points.

Moreover, in the current thesis work input data contains trajectories extracted from video cameras without sorting and analyzing, so:

1. input data set can contain both examples of normal and anomalous trajectories;

2. input data does not contain labels.

Aforementioned limitations lead to the necessity to use unsupervised methods to automatically extract normality and abnormality rules from unlabeled data [18].

Chapter 3

Background and State of the Art

3.1 Anomalies Detection Techniques

For solving the main task of detecting outliers and abnormalities in video traffic data different techniques are presented nowadays, and these approaches can be classified in various ways. For example, anomaly detection techniques as well as clustering approaches can be classified as supervised, semi-supervised and unsupervised on the grounds of the manner of labeling the input data] [15]:

1. *Supervised*. Input data used for training contains labels for both normal and anomalous instances. As a result the algorithm can build models for both normal and abnormal classes;

2. *Semi-Supervised [15]* or *Weakly-Supervised [7]*. Input training data set contains class labels only for normal data instances. Such techniques are more widely used than supervised approaches, since anomalous data instances are usually not predictable and random and it is difficult to provide examples to cover all possible anomalous events;

3. *Unsupervised*. Does not require input data to be labeled neither for normal nor anomalous data. Such algorithms are based on the expectation that normal data instances are significantly more frequent than anomalous ones in the test data set and therefore are not applicable when this assumption is disrupted.

Alternatively, according to surveys done by Chandola in [15] and Kumaran in [17], anomaly detection techniques can be classification based, nearest-neighbor based, clustering based, statistical and etc. The following offers the short overview of mentioned groups.

Classification based

The main concept of these methods lies in using a classifier which firstly learns to distinguish inliers and outliers and then classifies each input instance [19]. Such techniques consist of training and testing phases. Training, or learning, phase supposes learning a classifier model from a training data set, containing labeled data instances. The learned classifier is then used to classify an input trajectory as normal or anomalous by assigning a class label in a testing phase.

Depending on how testing data instances are labeled, all classification based anomaly detection techniques can be one-class or multi-class. The first type assumes that all training data instances are normal and are labeled as one class. During training phase model learns a discriminative boundary around normal instances, and a trajectory, which is not aligned with the learned normal class description, is considered as an anomalous. Single-class Support Vector Machines (SVMs) is the most commonly used classification based approach, which is applicable to the task of anomalous trajectory detection, as it was proposed by Piciarelli *et al.* [20][21]. However, this approach requires trajectory vectors to be the same length. Since raw trajectory data is usually contains different amount of trajectory points due to different speed of moving objects, it is necessary to preprocess raw trajectories to normalize them to vectors of the same length [21]. Moreover, SVMs become highly time and memory consuming while working with huge amounts of multi-dimensional data [22].

The latter category supposes learning multiple classes during training step and then using a classifier to review the input trajectory for compliance with each learned class. In literature different descriptions of training phase and training data labels are given. According to [15], training data contains only normal data instances with corresponding normal class labels, and during training phase model learns multiple discriminative boundaries around each class of normal instances.

A trajectory, which is aligned with none of the learned normal class descriptions, is considered as an anomalous. In other words, an anomalous trajectory will not be accepted by neither of the classifiers. In [17] it is assumed that model is learned using training data containing labels for normal and anomalous classes. Therefore, a classifier can classify an input trajectory as belonging to a normal or anomalous class.

The advantage of two-phased classification based algorithms is a fast testing phase due to precomputed classifier model used to classify each input instance. Also such algorithms can perform well in cases when anomalous data instances form a class or cluster [19]. However, the training step requires accurately labeled training data, which is often not available.

Nearest-neighbor based [15] or Proximity / Distance based [17][19]

Proximity based approaches decide whether a data instance is normal or anomalous based on how close or far is it located with respect to neighbors [17]. Nearest-neighbor and density based approaches are based on the assumption, that «normal data instances have dense neighborhood, while anomalous data instances occur far from their closest neighbors» [15].

In order to be able to compare the surrounding density for an instance under consideration with the density around its local neighbors, a distance (dissimilarity) or similarity measure between two data instances needs to be specified [19]. By virtue of an anomaly score calculation method, techniques can be grouped into two categories: 1) the anomaly score is calculated as a distance of a data instance to its k^th nearest neighbor and 2) to compute the anomaly score the relative density of each data instance is being computed [15].

These approaches has several disadvantages. First of all, in comparison with classification based anomalies detection techniques, the computational complexity of the testing phase is considerably higher, since nearest neighbors are computed by computing the distance for each test data instance with all instances from either testing and training data. In case of multi-dimensional trajectory data, the task of distance computation becomes even more complicated. Moreover, the accuracy of

labeling decreases when the main assumption is violated: when normal instances have sparse neighborhood or anomalous instances have dense [15].

Clustering based

Clustering is an efficient approach aimed to group data instances into different classes, called clusters, based on their similarity in such a way, that objects in one cluster are similar to each other and dissimilar to objects in other clusters [11][22]. ST clustering supposes grouping objects on the ground of their spatial and temporal similarity. To compare data instances before grouping them into clusters, similarity or distance between them needs to be measured.

There are three types of clustering based anomalies detection techniques with following assumptions: 1) normal data instances are associated with a cluster, while anomalous data instances are not associated with any cluster, 2) normal data instances are close to the cluster center, while abnormal instances lie far away from the closest cluster center and 3) normal data instances lie in large and dense clusters, while anomalies are associated with sparse clusters or clusters with a small cardinality [15][17]. Techniques of first type can be implemented using one of the clustering methods which do not require every data instance to belong to some cluster, for example DBSCAN [23]. Algorithms from second group consist of two phases: 1) data clustering and 2) calculating an anomaly score for each data instance. Techniques of the latter type require a threshold for cardinality size and/or density of a cluster to be defined to decide whether a cluster refers to normal or anomalous data.

The necessity to compute distance between trajectories in some of the clustering based approaches makes them similar to neighbor based approaches. As it is stated in [15], techniques are different in the way they process instances: in clustering based techniques each instance is evaluated with respect to the corresponding cluster, while in neighbor based techniques each instance is being inspected with respect to its proximate neighborhood. Consequently, the selection of distance computation method plays an important role and affects results and performance significantly.

On the other side, dividing all training data into groups makes clustering based algorithms similar to classification based algorithms. Though in classification based approaches class is assigned based on given labels, while in clustering based approaches classification is not given in advance [19].

One of the main advantages of clustering based techniques is the ability of majority of them to run in an unsupervised manner. For the case of TVS-based trajectory data acquisition the unsupervised learning methods are the most appropriate, because labeling hours of video data is a highly time-consuming task. Also, manual labeling of input data can lead to errors due to human operator intervention.

Moreover, clustering based techniques are adjustable to work with complex data types because of adaptability of clustering algorithms. However, at the same time they are computationally expensive, highly dependent on the used clustering algorithm and can not effectively deal with situations when anomalies form significant separate cluster groups [15].

Model based [17][19] or statistical [15]

The main concept of model based algorithms is that they represent the data as a set of parameters to create the model of a normal behavior. Statistics based approaches can be considered as a subcategory of model based approaches. As it is stated in [15], the main idea of statistical approaches is that data instances occurring in high probability regions of a stochastic model assumed to be normal, while data instances from the low probability regions refer to anomalies. So, statistical approaches are based on using statistical stochastic model to fit to the given data and then applying a statistical inference test, also called discordance test, to decide if a data instance is normal or anomalous. It comes from the main concept that «based on results of applied statistical test, anomalies have low probability to be generated from the learned stochastic model» [15].

Statistical techniques in turn can be parametric or non-parametric. In parametric approaches the normal data is supposed to fit the parametric distribution and probability density function with estimated from the given data parameters [17]. However, it is difficult to fit the data to one distribution. In this case it is

possible to use a multiple-distribution model to match some clusters of the data with particular distributions [19]. By contrast to this, non-parametric approaches are based on using non-parametric statistical models with structures, which are not defined in advance: the given data is used to determine the structure dynamically.

Since statistical approaches are based on fitting a statistical model, the choice of it significantly affects results, computational complexity and performance. Nevertheless, the main assumption of statistical approaches that the data comes from a particular distribution can not be always satisfied, specifically for the case of a multi-dimensional data [15].

Summary

Based on the given description of different approaches and their advantages and disadvantages, it was decided to focus on clustering based anomalies detection approaches for several reasons:

1) they can work in an unsupervised mode without a human intervention and do not require the input data to contain labels,

2) input data is allowed to contain anomalous trajectories,

3) clustering method can be easily applied to such a multi-dimensional data as trajectories by defining a suitable similarity measure.

That means that a clustering method and a similarity measure need to be specified.

3.2 Clustering Approaches

Clustering is a highly researched form of data mining, and huge variety of clustering methods has already been proposed in literature [11]. State-of-the-art analysis of related research papers revealed that all traditional clustering approaches are usually categorized into five types: partitioning, hierarchical, density-based, model-based and grid-based methods [7][11]. Next paragraphs will briefly discuss each of the categories with highlighting main assumptions and concepts.

Partitioning, or Partition-based, methods

Such methods are based on partitioning the trajectories data set randomly and then regrouping clusters by reassigning objects from one partition to another to minimize the objective function. They require the predefined parameter, usually denoted as k, which determines the amount of final clusters, or partitions, to be created. The main requirement is that number of partitions must be smaller than number of initial data points, since each partition forms a cluster, that means that it must be non-empty and contain at least one data instance, and each data instance must be included into exactly one cluster.

One of the most well-known examples of partitioning clustering algorithms is a K-Means algorithm, where firstly k cluster centers are initialized randomly and then data points are iteratively reassigned to the closest clustering center based on the discrepancy to minimize the clustering error [24]. The clustering error is defined as the sum of the squared Euclidean distances between each data set point and the corresponding cluster center [25]. The process is stopped when there are no more changes in clustering centers.

The disadvantages of the traditional K-Means clustering method are inability to form clusters of arbitrary form, dependence on initial random cluster centers initialization and high memory consumption [11]. Also finding an appropriate partitioning technique is a challenging task.

Hierarchical methods

In hierarchical based methods the given data set is decomposed into multiple levels to organize a hierarchical tree of clusters. The resulting hierarchical structure can be depicted as a tree [24].

There are two different ways of hierarchical decomposition: 1) the bottom-up (combining) and 2) the top-down (split, divisive) decomposition. They refer to agglomerative and divisive (split) clustering approaches respectively. Agglomerative hierarchical clustering algorithms start by assigning each data instance to a distinct singleton-cluster, so the number of initial clusters is equal to the exact amount of data instances in input data, and then continue uniting clusters based on theirs similarity. Proximity matrix is used to store similarity

measurements between clusters and is being updated on each step by computing distances between the new cluster and the other clusters. The divisive hierarchical clustering algorithms work in a reverse manner: initially all data instances belong to one cluster and then step by step clusters split into smaller clusters until all of them become singleton clusters or until satisfying some predefined end condition.

Hierarchical clustering is supposed to be simple, but it is necessary to choose between agglomerative and split methods. Divisive clustering is more expensive in computation, therefore, it is less common than agglomerative approaches. Irreversibility of both splitting or uniting processes in traditional hierarchical clustering algorithms is also a particularity of such algorithms [11].

Since approach includes clusters joining, a significant task of agglomerative clustering algorithms is defining and computing the similarity or distance between clusters. This similarity can also be referred to as an inter-cluster or between-cluster distance. For the case of single-trajectory clusters the similarity between them is simplified and is equal to the similarity between respective trajectories. For multiple-trajectory clusters the similarity is computed according to a chosen linkage method. In literature following linkage methods are given as mostly common: single link, complete link, average link [24][26]. In the case of the single link distance between two clusters is defined as the minimum distance between two trajectories in these clusters, that means that the similarity between of two clusters is determined by two closest trajectories. The complete link linkage method implies taking the maximum distance between two trajectories in two clusters as an inter-cluster distance, so it is defined using the farthest distance of trajectory pairs. The average link supposes calculating averaged paired distance between all trajectory pairs in these two clusters.

A convenience of agglomerative hierarchical clustering approaches is that they do not require the number of resulting clusters to be predefined, so they are appropriate for clustering vehicle trajectories, because number of clusters of normal or anomalous trajectories is not known in advance. However, the most well-known disadvantage of hierarchical clustering algorithms is that they are not robust and can suffer from noise and anomalies.

Density-based methods

In comparison with the partitioning and hierarchical clustering approaches, density-based methods objects inspect similarity based on the density of the data [22]. The area is being added to the nearest cluster, while density of the points in the area remains greater than the predefined threshold [11]. Clusters form dense regions of objects and they are separated by sparse regions with low density.

The main advantage of density-based clustering approaches is that they are able to form clusters of arbitrary forms, extend beyond spherical [11]. Also they are appropriate for clustering huge data sets of trajectories in an unsupervised manner and do not require the amount of clusters to be known in advance [7][22]. However, the results quality highly dependent on the amount of trajectories in training data set, available for analysis.

The most well-known and commonly used density-based algorithm is a DBSCAN, proposed by M. Ester *et al.* in [23]. According to it, input data points are categorized as follows: core data, density-reachable data and outliers based on parameters ε, *minPts* and the density threshold. Neighbor parameter ε and *minPts* specify the maximum remoteness and minimum amount of satisfying points while choosing the core points: at least *minPts* points must be present within distance ε from the core point, these points are marked as directly reachable from the chosen core point. Aforementioned parameters need to be predefined by the user, but it is difficult to determine them correctly. Each cluster must contain at least one core point. Points are denoted as anomalous if they are not reachable from any of the other points.

Shrinkage-based or Grid-based methods

The main idea of grid-based algorithms lies in applying a multi-resolution grid data structure: the data space is quantized into a finite number of cells (units) that form a multi-resolutional grid structure. Each cell stores summary information about data objects within its subspace [22]. Since clustering operations are performed on the created grid, and also important trajectories characteristics can be computed in each of the spatial grid cells, the quality of data compression influences the quality of results significantly [8]. Density of closely located

dense cells can help to determine clusters. A trajectory can be considered as an anomalous if it differs from the expected trajectory in a number of covered grid cells [22].

The main advantage of grid-based clustering algorithms is an improved performance: increased processing speed and processing time becomes independent on the size of the data set, only the number of cells in each dimension in the quantized space affects the processing time [11].

Model-based methods

In comparison with the above methods, which analyze distance among data objects, in model-based approaches data is supposed to be generated by a mixture of probability distributions, where each component of mixture represents a cluster. So a mathematical model is assigned to each cluster, and then method attempts to find the best fitting data for the chosen model. In this way such methods seek to increase the adaptability between given data and some statistical models [11][22]. The idea of model-based algorithms is that in order to locate clusters they describe the spatial distribution of the input data points by building density functions. The model-based approaches are typically used in feature-specific clustering and depend on the selected features and model [7].

It is emphasized, that model-based approaches show good performance while working with complex data types. This category usually includes statistical and neural network methods [11].

Graph-based methods [8]

Another category of clustering methods in application to vehicle trajectories data. Liu *et al.* in [27] presented a graph-based approach to solve the problem of detection of outliers in traffic data streams. A graph structure was used to store the traffic: nodes represent regions while edge weights depict the traffic flow. Edge anomalies in the graph denote the traffic abnormalities, and causal outlier tree can then be used to further analyze these outliers to find causal interactions.

Another higher-level classification of clustering methods can consist of only two sub-classes on the ground of properties of generated clusters: hierarchical and

partitioning approaches [24]. Hierarchical algorithms group objects into clusters from singleton cluster to cluster containing all data instances or in a reverse direction. While partitioning clustering algorithms divide given data set into a predefined number of clusters in a single-layer structure.

In order to perform clustering, the similarity between two trajectories needs to be defined. Different existing distance measures will be reviewed in following paragraphs.

3.3 Distance and Similarity Measures

As it was mentioned before, clustering based approaches require a similarity measure to be defined between two trajectories. Apart from that, distance and similarity measures are also used to compare a trajectory with a cluster or a pair of clusters between each other. A similarity measure highly dependent on the format of a trajectory. A trajectory data, represented as a multidimensional data, can contain quantitative or qualitative features, continuous or binary. In such a classification, distance measure functions are more appropriate to work with continuous features, while similarity measures – with qualitative features [24]. Input trajectory-vectors in this work contain spatial information along with temporal, which can be termed as qualitative continuous data. That means that distance measure functions are more appropriate in this case. Moreover, distance and similarity functions can be classified as 1) working with raw representations of trajectories without any preprocessing steps and 2) working with preprocessed trajectories representations. Preprocessing can include unifying the length of trajectories or reducing the dimensionality of trajectory-vectors [26].

Some of the most known, widely used traditional similarity measures are following: Euclidean distance, Fréchet Distance, DTW, LCSS.

Euclidean distance

Euclidean distance between two trajectory vectors is calculated as a sum of squared differences of corresponding spatial coordinates [18]:

$$d_{ij} = ||T_i - T_j||_E = \sqrt{\sum_{k=1}^{m}((t_{i_x}^k - t_{i_x}^k)^2 + (t_{i_y}^k - t_{j_y}^k)^2)}, \tag{3.3.1}$$

where both trajectories consist of *m* tracking points and are represented by two-dimensional vectors $T_i = \{t_i^1, t_i^2, \ldots, t_i^m\}$ and $T_j = \{t_j^1, t_j^2, \ldots, t_j^m\}$. Tuples $(t_{i_x}^k, t_{i_y}^k)$ represent spatial coordinates for a *k*-th tracking point of *i*-th trajectory from a data set.

However, Euclidean distance works only with trajectories with equal number of tracking points. Since usually vehicles move with different speed and behavior, trajectory length is always different and that means that raw trajectories need to be preprocessed and reduced to the same size [26]. Also, traditional Euclidean distance requires two-dimensional data, meaning that it is not able to process temporal information, and is dependent on the trajectory direction: the reversed direction can cause incorrect distance measurement, that in its turn leads to errors in clustering. Also, it fails while working with trajectories moving in a similar way but with different speeds and in the case of different sampling rates [28].

Fréchet Distance

Fréchet Distance is based on Euclidean distance. It considers the positional and sequential relationship of trajectory points while calculating the similarity. The main idea of this approach is computing Euclidean distance for each pair of points from two trajectories and then designating the maximum Euclidean distance as a Fréchet Distance between them [11][29]. However, since only the maximum among distance is considered, the approach is sensitive to the presence of outliers.

DTW

Dynamic Time Warping (DTW) is one of the algorithms for measuring the similarity between two temporal time series sequences, which may vary in speed.

The objective of time series comparison methods is to produce a distance metric between them two. DTW method aims to find an alignment between time-dependent sequences, such as trajectories, and is able to process trajectories of different lengths [11].

According to [11], DTW distance is calculated as follows (Formula 3.3.2):

$$D_D(T_i, T_j) = \begin{cases} 0 & \text{m = n = 0} \\ \infty & \text{m = 0 || n = 0} \\ dist(a_i^k, b_j^k) + min \begin{cases} D_D(Rest(T_i), Rest(T_j)) \\ D_D(Rest(T_i), T_j) \\ D_D(T_i, Rest(T_j)) \end{cases} & \text{others} \end{cases} \tag{3.3.2}$$

where $D_D(T_i, T_j)$ refers to DTW distance between two trajectory segments with lengths m and n, $dist(a_i, b_j)$ means the Euclidean Distance between two trajectory points. Function $Rest(T_i)$ takes the remaining part of a trajectory after excluding the point a_i. It can be seen, that in case of zero-length trajectories the DTW distance is equal to 0, for the case then only one of two trajectories is non-empty, the distance between them is considered to be infinite. For two non-empty trajectories, the minimum distance between them is calculated in a recursive way.

Though the important advantage of the DTW method is its ability to process trajectory vectors of distinct lengths, DTW distance is not robust to noise and requires trajectory points to be continuous. Also DTW distance computation is highly time consuming and complex due to necessity to compare distances between each pair of trajectories.

LCSS

Longest Common SubSequence (LCSS) distance tries to match two trajectory sequences based on the longest common sub-sequence between them. The task of finding the longest common sub-sequence is usually solved recursively [11]. The basic idea of an LCSS distance is that it allows two trajectories to stretch. In

comparison with DTW and Euclidean distances, LCSS enables some elements to remain unmatched.

The LCSS distance is calculated according to the Formula 3.3.3 [26]:

$$D_{LCSS}(T_1, T_2) = 1 - LCSS_{\delta,\epsilon}(T_1, T_2)/min(m, n) \tag{3.3.3}$$

where m and n are lengths of trajectories T_1 and T_2 respectively. The $LCSS_{\delta,\epsilon}(T_1, T_2)$, the longest common sub-sequence between trajectories, represents the number of matched trajectory points between trajectories T_1 and T_2 and is defined as follows (Formula 3.3.4):

$$LCSS_{\delta,\epsilon}(T_1, T_2) = \begin{cases} 0 & \text{if m = 0 or n = 0} \\ 1 + LCSS_{\delta,\epsilon}(Rest(T_1), Rest(T_2)) & \text{(if } |t_{1_{x,m}} - t_{2_{x,n}}| < \epsilon \text{ and } |t_{1_{y,m}} - t_{2_{y,n}}| < \epsilon \text{ and } |m - n| \leq \delta) \\ max \begin{cases} LCSS_{\delta,\epsilon}(Rest(T_1), T_2) \\ LCSS_{\delta,\epsilon}(T_1, Rest(T_2)) \end{cases} & \text{otherwise} \end{cases} \tag{3.3.4}$$

As it can be seen, LCSS calculation depends on two constant parameters: δ and ϵ. Parameter δ defines *the maximum remoteness in terms of time between two trajectory points in which we can look to match a given point from one trajectory with another*. Constant ϵ can take value between 0 and 1 and defines the size of proximity to look for matches in terms of spatial information. Difference between $X-$ and $Y-$coordinates less than ϵ value means that points are relatively close to each other and can be considered as similar. LCSS distance is increased by 1 in this case. Parameters δ and ϵ affect results significantly, therefore, the task of choosing the optimal values for them is challenging and important [26][28]. The $Rest(T)$ function is defined to return the last $M - 1$ points from the trajectory T.

The LCSS distance is the most appropriate in this work, since it allows the trajectories to contain noise, have different length, objects speed and sampling rates (local time shifts in trajectories) [26]. Moreover, among the aforementioned methods, the LCSS distance is the most robust approach against noises.

3.4 Related Work

The aforementioned objective has been investigated and solved in numerous works using different methods. Since in fact normal events are common and dominate the data, and abnormal events are rare and difficult to describe explicitly, many approaches are based on an unsupervised clustering of trajectories. For this thesis work the approach proposed by Ghrab, Fendri, Hammami in [26] was chosen as a basis. It is focused on detection of abnormalities based on a trajectories clustering.

The proposed approach consists of two phases (3.4.1):

- *offline* to perform clustering and extract frequent trajectories, and
- *online* to classify the new trajectory as a normal or abnormal one.

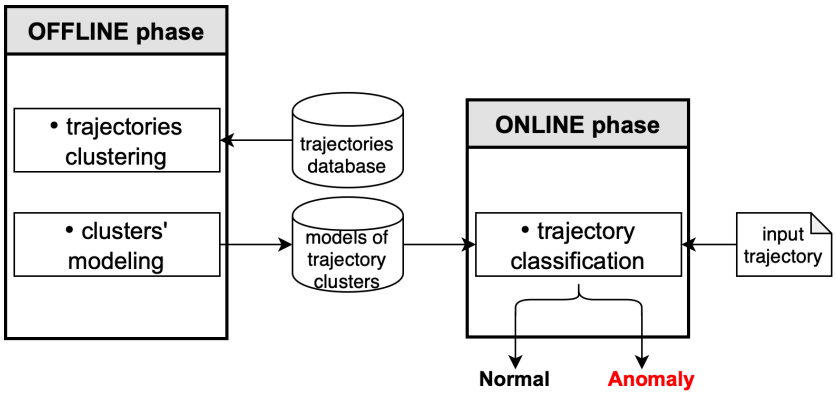

Figure 3.4.1: Two-phased proposed approach

The clustering is done in an unsupervised manner using an agglomerative hierarchical clustering algorithm operating on a distance matrix between trajectories. To perform clustering, the LCSS distance is used as a similarity measure. The formulas and description of the LCSS distance are given in the previous section (Formulas 3.3.3 – 3.3.4).

As it was already mentioned, hierarchical agglomerative clustering methods suppose clusters joining, which requires the inter-cluster distance measure to be defined. In [26] authors have performed evaluation of different linkage methods, including single link, complete link and average link. According to the performed

tests, the single link method showed the best results and, in view of this, will be used as a linkage method in the current work.

Single link linkage method considers a minimum distance between two trajectories as an inter-cluster distance and can be summed up as:

$$D_{min}(C_i, C_j) = \min_{T_1 \in C_i, T_2 \in C_j} D_{LCSS}(T_1, T_2), \tag{3.4.1}$$

where (C_i, C_j) denote two clusters and (T_1, T_2) correspond to two trajectories from two clusters respectively.

Advantages

One of the advantages of the proposed method is that the chosen similarity measure does not require the trajectories to be of the same length, so that the preprocessing of the trajectories, which is a high complexity process, can be avoided. Moreover, the training data is allowed to contain normal trajectories as well as anomalous: algorithm will extract both normal and anomalous clusters. Dense clusters will represent normal trajectories classes, sparse clusters – anomalous trajectories classes.

Disadvantages

However, the disadvantage of the proposed method is that LCSS distance does not take into consideration such problems of video surveillance as a view perspective and a position of a moving object.

Summary

This thesis work will be intended to investigate an opportunity of increasing the accuracy of results by making epsilon and sigma parameters, which are used to calculate the sigma, adaptable and dependent on the perspective and a distance from the camera. This includes:

- exploring a functional dependency between epsilon and sigma parameters and a distance from the camera,

- evaluating algorithm with different values.

3.5 Big Data Processing Toolkits

3.5.1 Apache Hadoop

Apache Hadoop is a well-known open source software framework written in Java, consisting of different libraries and tools, aimed to handle big data in a clustered file system and focused on a distributed computing [30].

The main parts of Apache Hadoop are the storage part, which is called Hadoop Distributed File System (HDFS) because of the distributed manner of storing data, and the data processing part based on a MapReduce programming model, which allows to scale up from a single server to multiple "worker"-machines, providing local storage and computing. This allows the distributed processing of large data sets across clusters of computers using simple programming models: the whole application can be divided into small tasks executable on each "worker"-node machine.

The framework is composed of 4 main modules which are designed to be resilient, fault-tolerant with an idea to detect and automatically handle failures on the application level [30]:

- *Hadoop Distributed File System (HDFS)*
- *Hadoop MapReduce*
- *Hadoop Yet Another Resource Negotiator (YARN)*
- *Hadoop Common*

HDFS

HDFS is a distributed file system for storing data on node-machines in a cluster by dividing the data file into a sequence of blocks of equal size (except the last block, which can have an arbitrary size). In order to provide reliability of stored data, HDFS implements data replication: the default replication degree is equal to 3, meaning storing the same block of data three times. However, the HDFS system is modeled to work with immutable files, all files are write-once and can

have only one writer at a time. That makes HDFS inappropriate for concurrent writing operations.

Hadoop MapReduce

Hadoop MapReduce is an implementation of the MapReduce programming model for parallel processing of large-scale data. The MapReduce job split the input data file into a number of independent blocks and then process them using two tasks:

- *map task* - processes independent blocks in a parallel manner by converting the data into a set of <*key-value*> pairs according to a defined function,
- *reduce task* - takes the outputs of a previous task from multiple nodes and combines them into a whole result using a defined reduce function.

The tasks scheduling and execution is being monitored by the framework and can be re-executed in the case of a failure.

Hadoop YARN

Hadoop YARN is a module for managing cluster computing resources and users' applications scheduling. Its main goal is to effectively allocate resources to multiple applications. YARN implements that by distinguishing the glogal Resource Manager, which is responsible for tracking jobs and allocation of resources to applications, and the per-application Application Master, which is responsible for monitoring the execution progress.

Hadoop Common

Hadoop Common is a package that provides file system and level abstractions for an operating system. It contains libraries and utilities necessary for other modules of Hadoop and scripts to run Hadoop.

Summary

However, since Apache Hadoop architecture is closely linked to HDFS, the big data processing requires a large number of read and write operations with data storage, which result in a slow performance.

3.5.2 Apache Spark

Apache Spark is an open source framework which was proposed as an analytics engine for processing of a large-scale data: data analytics, machine learning algorithms, cluster computing [31]. It is compatible with a wide range of programming languages, such as Scala, Java, Python and R.

One of the main advantages of Apache Spark is that interactions with a storage are only needed to load the input data and write the output results. In comparison with Apache Hadoop, Apache Spark stores data in a RAM of each node instead of a disk. It was presumed to provide more efficient processing for distributed data than Apache Hadoop for the reason of in-memory cluster computing. Apache Spark has been found to run 100 times faster in-memory, and 10 times faster on disk [32].

The main concept of Spark is using a resilient computational model. In earlier versions of Apache Spark (before 2.0.0) Resilient Distributed Dataset (RDD) was used a main structure to work with data. In Spark 2.0.0 Dataset structure was introduced, which provides richer optimizations but remains strongly-typed as RDD [31]. RDD is a read-only structure and looks like a multiset of data items of different types distributed between cluster-nodes, while each node is maintained in a fault-tolerant way. That makes an RDD data structure fault-tolerant and parallel. RDDs can be created by loading an external file from disk or any other resource or from existing RDDs as a result of methods called from the collection elements (action-type operations). RDD can also be created by a parallelization of a common array.

All available on RDD operations can be divided into two categories [33]:

- *transformations* - operations that do not produce any output, as a result they convert RDD to a new RDD by applying specified calculations. By

creating a new RDD they maintain the principle of objects' immutability. Transformations are performed in a lazy mode: they will not be executed immediately but only after invoking one of the action operations. Examples of transformations are $map()$, $filter()$, $join()$ functions and etc.;

- *actions* - operations that produce an output value to return to the application or to export data to a storage system. Examples of actions are $count()$, $collect()$ functions.

On top of Spark Core, Apache Spark provides several extensions for different data mining tasks [31]:

- *Spark SQL and DataFrames* - is a module for working with structured data. In comparison with the classic Spark RDD API, the Spark SQL interfaces give more information about the structure of the data and the performed computation. One of the applications is adding a support for SQL language and providing an ability to execute SQL-queries on the data;

- *Spark Streaming* - is a component of Spark to build scalable, fault-tolerant, high-throughput streaming and interactive applications. It allows to apply algorithms from other Spark extensions, such as Spark MLlib and GraphX, to data streams. Under the hood, after receiving the live input data stream, Spark Streaming divides the data into batches and then processes them by the Spark engine, which generates the batches of the final results stream;

- *Spark MLlib* - is a scalable library for machine learning. It provides tools for working with common machine learning algorithms, such as classification, clustering, regression, for working with machine learning pipelines, for featurization the data, meaning extraction the features, reduction the dimensionality or transformation, and utilities for linear algebra and statistics.

- *GraphX* - is an extension adding the support for graph data structure and computations on graphs in a distributed manner by creating a new $Graph$ abstraction. The extension contains a collection of graph algorithms and builders, simplifying the tasks of graph analysis.

3.5.3 STARK

Notwithstanding that Apache Spark has rich environment, it does not natively support operations on spatial or spatio-temporal data. The necessity to introduce custom classes to store the spatial and/or temporal information and to implement operations on them is challenging and inefficient. Moreover, since Apache Spark is designed to work with large volumes of data and spatio-temporal information, generated by input sources, also usually relate to big data, it is important to perform partitioning the data in an appropriate way. For that reason, several extensions providing spatial data support were introduced, such as GeoSpark [34][35] or SpatialSpark [36]. However, they still do not work with temporal data.

In 2017 in [1] authors proposed a STARK framework, which is written in Scala and adds a support for spatio-temporal data types and operations. To improve the efficiency of computations, STARK implements spatial partitioning and indexing.

STARK Architecture

STARK provides seamless integration of its functionality into any Apache Spark application and uses features of Scala language. That makes STARK's functions intuitive and self-explanatory for users who are familiar with RDD API. Figure 3.5.1 depicts an overview of the STARK architecture and integration of STARK into Spark framework.

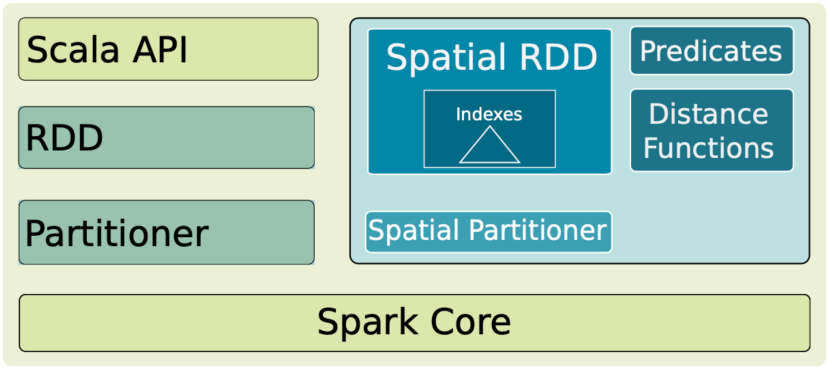

Figure 3.5.1: Overview of STARK architecture and integration into Apache Spark [1]

To add support of spatio-temporal operators to standard RDDs and to work with spatio-temporal data, STARK implements following classes: STObject, SpatialRDD. STObject is a basic data structure in STARK. It is used to represent the spatial or spatio-temporal component of any object and has two fields: 1) geo to store the spatial data and 2) $time$ to store the temporal component. Also STObject class provides operations to test relations between instances: $intersect(o)$, $contains(o)$, $containedBy(o)$. A SpatialRDD is intended to store spatio-temporal vector data sets. Figure 3.5.2 shows the architecture of STARK framework with emphasizing added functionality.

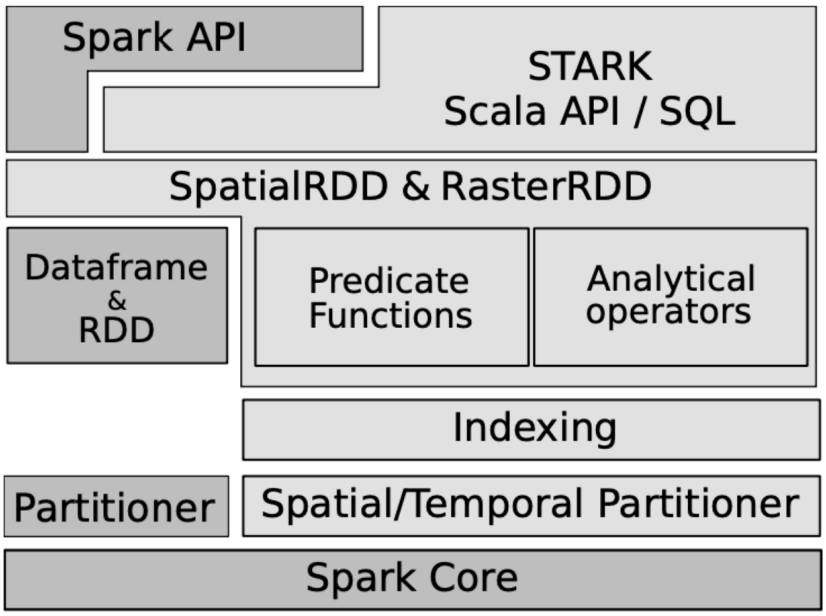

Figure 3.5.2: Detailed STARK architecture [2]

3.5.4 Summary

According to the aforegoing description, it was decided to use following technologies for implementation part of the thesis work:

- Apache Spark is chosen as a base platform for implementation, since it offers higher performance in comparison with Apache Hadoop and has powerful extensions.

- STARK framework is chosen to perform operations on spatio-temporal trajectory data, since Apache Spark natively lacks support for spatio-temporal data processing.

Chapter 4

Framework

In this chapter the description of the framework development is given. In first sections the conceptual model and the architecture of the solution are discussed. Further sections provide the details of the solution implementation.

4.1 Framework Conceptual Model

4.2 Framework Architecture

4.3 Framework Implementation

4.3.1 Input Data Description (Nature of Data)

According to the research done by the US Department of Transportation based on data of Fatality Analysis Reporting System (FARS) and National Automotive Sampling System, nearly 40 percents of all the reported in 2008 year crashes were road intersection related [37]. Consequently, cross-road transport activity analysis is significantly important nowadays in context of safety, and identifying unsafe vehicular trajectories, which violate traffic rules, may be one of the steps towards improving the statistics.

In the presented work video from enforcement cameras is used for training and testing. Test videos are captured using the Intellectual Transportation Systems implemented on four different Kazan crossroads:

1. An intersection of Pravo-Bulachnaya and Puschkina streets (Figure 4.3.1).
2. An intersection of Nesmelova and Kirovskaya Damba streets (Figure 4.3.2).
3. An intersection of Moskovskaya and Galiaskara Kamala streets (Figure 4.3.3).
4. An intersection of Moskovskaya and Parizhskoy Kommunyi streets (Figure 4.3.4).

Each crossroad corresponds to a 4-way intersection and is equipped with a single monitoring camera. Sample pictures from surveillance cameras are given below on Figures 4.3.1 – 4.3.4.

Figure 4.3.1: Pravo-Bulachnaya / Puschkina intersection

Input data files contain 624, 211, 231, 237 vehicular trajectories for the each of the aforementioned intersections respectively.

By a trajectory anomaly we understand vehicle trajectories through the crossroad, which remarkably differ from majority of common, known trajectories. For example, if no turning to the right from the left line is allowed, such a behavior will be unknown and such a trajectory must be considered as an anomaly.

Figure 4.3.2: Nesmelova / Kirovskaya Damba intersection

Figure 4.3.3: Moskovskaya / Galiaskara Kamala intersection

Input data file structure

Tracking system, as it was described before, handles video from enforcement cameras and prepare it for further analysis: converts video stream into a set of vectors with tracking points on images (Figure 4.3.5).

Input data files have the following structure:

$$[[[(x_1^1, y_1^1), ..., (x_1^n, y_1^n)], [t_1, ...t_n]], [[(x_2^1, y_2^1), ..., (x_2^m, y_2^m)], [t_1, ...t_m]], ...] \quad (4.3.1)$$

As it can be seen from the input data file structure, each trajectory is represented by a two-element array, where first array stores coordinates as an array of

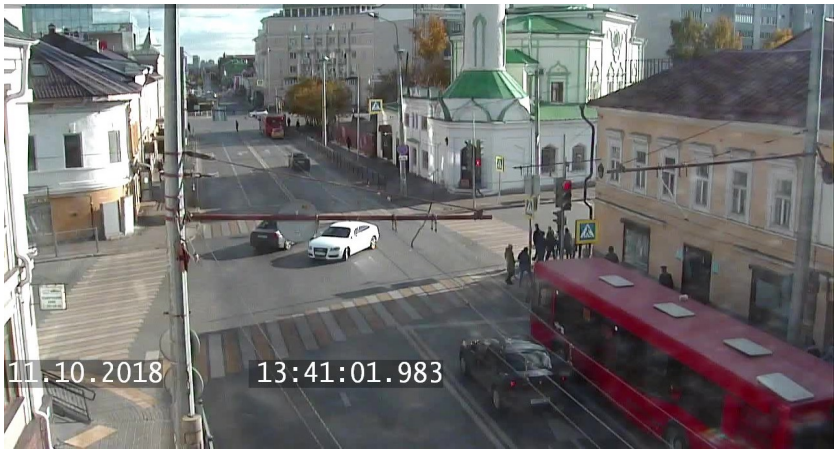

Figure 4.3.4: Moskovskaya / Parizhskoy Kommunyi intersection

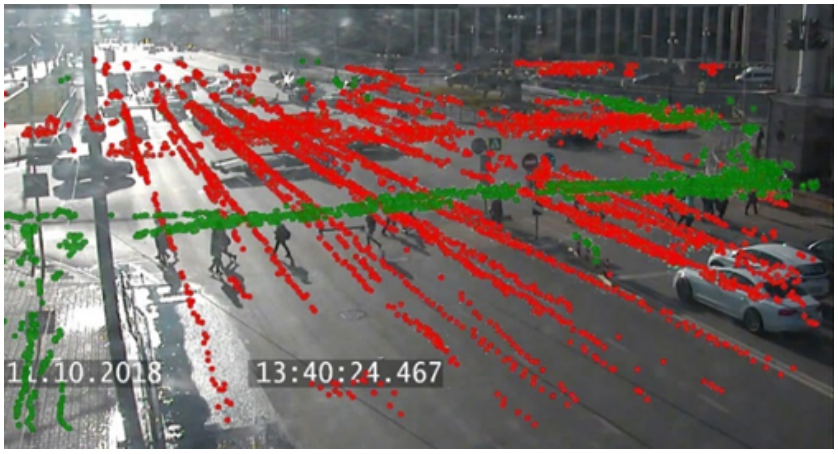

Figure 4.3.5: Output of a tracking system for video the first intersection

two-tuples (x_i^j, y_i^j) and second array contains timestamps for each spatial point in the corresponding order (t_i). The extracted x- and y-coordinates correspond to pixels on input images. In Formula 4.3.1 the lower index of the spatial coordinates indicates the ordering number of a trajectory, while the upper index indicates the ordering number of a tracking point. The outer array refers to the array of trajectories.

4.3.2 Input Data Processing

Since chosen algorithm requires trajectories in a form of multi-dimensional vectors, the initial input data needs to be converted into the required form. For that reason, a custom parser was implemented. It takes a 'txt' file with trajectories as an input and as a result it returns a list of Trajectory objects. Trajectory object consists of a number of TrajectoryPoint objects with following information: x-coordinate, y-coordinate, time t. The source code of the parsing method is presented in the Listing in Appendix chapter.

Chapter 5

Evaluation & Results

Chapter 6

Conclusion & Perspectives

In this work following results were achieved:

- task1,
- task2.

The implemented algorithm is designed in an offline-learning manner, that means that models of normal trajectories are learned offline beforehand and are not updated with new upcoming data on an on-going basis. The future researches can include investigating an opportunity of updating normal trajectories database in order to make the framework more adaptable to actual traffic data.

Bibliography

[1] S. Hagedorn, P. Götze, and K.-U. Sattler. The STARK Framework for Spatio-Temporal Data Analytics on Spark. *Datenbanksysteme für Business, Technologie und Web (BTW 2017), Gesellschaft für Informatik, Bonn*, pages 123–142, 2017.

[2] S. Hagedorn, T. Räth, O. Birli, and K.-U. Sattler. Processing Large Raster and Vector Data in Apache Spark. *Datenbanksysteme für Business, Technologie und Web (BTW 2019), Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn*, pages 551–554, 2019.

[3] Y. Djenouri, A. Belhadi, J. C. Lin, D. Djenouri, and A. Cano. A Survey on Urban Traffic Anomalies Detection Algorithms. *IEEE Access*, 7:12192–12205, 2019.

[4] F. Mehboob, M. Abbas, R. Jiang, A. Rauf, S. A. Khan, and S. Rehman. Trajectory Based Vehicle Counting and Anomalous Event Visualization in Smart Cities. *Cluster Computing*, 21:443–452, March 2018.

[5] C. Koetsier, S. Busch, and M. Sester. Trajectory Extraction for Analysis of Unsafe Driving Behaviour. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42(2/W13):1573–1578, June 2019.

[6] R. Ranjith, J. J. Athanesious, and V. Vaidehi. Anomaly Detection using DBSCAN Clustering Technique for Traffic Video Surveillance. In *2015 7th International Conference on Advanced Computing (ICoAC)*, pages 1–6, December 2015.

[7] S. A. Ahmed, D. P. Dogra, S. Kar, and P. P. Roy. Trajectory-Based Surveillance Analysis: A Survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(7):1985–1997, 2019.

[8] G. Atluri, A. Karpatne, and V. Kumar. Spatio-Temporal Data Mining: A Survey of Problems and Methods. *ACM Computing Surveys*, 51(4), 2017.

[9] F. Tung, J. S. Zelek, and D. A. Clausi. Goal-Based Trajectory Analysis for Unusual Behaviour Detection in Intelligent Surveillance. *Image Vision Comput.*, 29(4):230–240, March 2011.

[10] Y. Li, J. Bailey, L. Kulik, and J. Pei. Mining Probabilistic Frequent Spatio-Temporal Sequential Patterns with Gap Constraints from Uncertain Databases. In *2013 IEEE 13th International Conference on Data Mining*, pages 448–457, December 2013.

[11] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang. A Review of Moving Object Trajectory Clustering Algorithms. *Artificial Intelligence Review*, 47(1):123–144, January 2017.

[12] A. d'Acierno, A. Saggese, and M. Vento. Designing Huge Repositories of Moving Vehicles Trajectories for Efficient Extraction of Semantic Data. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):2038–2049, August 2015.

[13] V. Bogorny V. C. Fontes. Discovering Semantic Spatial and Spatio-Temporal Outliers from Moving Object Trajectories. *ArXiv*, abs/1303.5132, 2013.

[14] H. Liu, X. Li, J. Li, and S. Zhang. Efficient Outlier Detection for High-Dimensional Data. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(12):2451–2461, December 2018.

[15] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys*, 41(3), July 2009.

[16] F. E. Grubbs. Procedures for Detecting Outlying Observations in Samples. *Technometrics*, 11(1):1–21, February 1969.

[17] S. K. Kumaran, D. P. Dogra, and P. P. Roy. Anomaly Detection in Road Traffic Using Visual Surveillance: A Survey. *ArXiv: Computer Vision and Pattern Recognition*, January 2019.

[18] D. Kumar, J. Bezdek, S. Rajasegarar, C. Leckie, and M. Palaniswami. A Visual-Numeric Approach to Clustering and Anomaly Detection for Trajectory Data. *The Visual Computer*, 33(3):265–281, March 2017.

[19] S. W. T. T. Liu, H. Y. T. Ngan, M. K. Ng, and S. J. Simske. Accumulated Relative Density Outlier Detection For Large Scale Traffic Data. In *Electronic Imaging*, volume 9, pages 1–10, 2018.

[20] P. Batapati, D. Tran, W. Sheng, M. Liu, and R. Zeng. Video Analysis for Traffic Anomaly Detection using Support Vector Machines. In *Proceedings of the 11th World Congress on Intelligent Control and Automation (WCICA)*, pages 5500–5505, March 2014.

[21] C. Piciarelli, C. Micheloni, and G. L. Foresti. Trajectory-Based Anomalous Event Detection. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1544–1554, December 2008.

[22] H.-L. Nguyen, Y.-K. Woon, and W. K. Ng. A Survey on Data Stream Clustering and Classification. *Knowledge and Information Systems*, 45:535–569, December 2014.

[23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[24] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

[25] G. F. Tzortzis and A. C. Likas. The Global Kernel k-Means Algorithm for Clustering in Feature Space. *IEEE Transactions on Neural Networks*, 20(7):1181–1194, July 2009.

[26] N. B. Ghrab, E. Fendri, and M. Hammami. Abnormal Events Detection Based on Trajectory Clustering. In *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*, pages 301–306, 2016.

[27] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing. Discovering Spatio-Temporal Causal Interactions in Traffic Data Streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1010–1018, New York, NY, USA, 2011. Association for Computing Machinery.

[28] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering Similar Multidimensional Trajectories. In *Proceedings 18th International Conference on Data Engineering*, pages 673–684, February 2002.

[29] T. Eiter and H. Mannila. Computing Discrete Fréchet Distance *. In *Technical report CD-TR 94/64, Technische Universitat Wien*, 1994.

[30] Apache Hadoop. `http://hadoop.apache.org/`. Internet Resource, Accessed: 2020-06-23.

[31] Apache Spark. `https://spark.apache.org/`. Internet Resource, Accessed: 2020-06-23.

[32] S. Goyal. Hadoop Vs Spark — Choosing the Right Big Data Framework. `https://www.netsolutions.com/insights/hadoop-vs-spark/`, May 2019. Internet Resource, Accessed: 2020-06-23.

[33] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 15–28, San Jose, CA, January 2012. USENIX.

[34] J. Yu, J. Wu, and M. Sarwat. A Demonstration of GeoSpark: A Cluster Computing Framework for Processing Big Spatial Data. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1410–1413, May 2016.

[35] J. Yu, J. Wu, and M. Sarwat. GeoSpark: A Cluster Computing Framework for Processing Large-Scale Spatial Sata. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, pages 1–4, New York, NY, USA, November 2015. Association for Computing Machinery.

[36] S. You, J. Zhang, and L. Gruenwald. Large-Scale Spatial Join Query Processing in Cloud. In *2015 31st IEEE International Conference on Data Engineering Workshops*, pages 34–41, 2015.

[37] E.-H. Choi and National Highway Traffic Safety Administration. Crash Factors in Intersection-Related Crashes: An On-Scene Perspective. In *NHTSA Technical Report DOT HS 811 366*, September 2010.

Todo list