Intelligent Video Surveillance

Dmitry Kangin

Supervisors: Prof. Plamen P. Angelov, PhD, DSc, FIEEE, FIET

Prof. Garegin Markarian, PhD, DSc, FIET

A thesis presented for the degree of

Doctor of Philosophy

Data Science Group

School of Computing and Communications

Lancaster University

England

February 2016

Abstract

In the focus of this thesis are the new and modified algorithms for object detection, recognition and tracking within the context of video analytics. The manual video surveillance has been proven to have low effectiveness and, at the same time, high expense because of the need in manual labour of operators, which are additionally prone to erroneous decisions. Along with increase of the number of surveillance cameras, there is a strong need to push for automatisation of the video analytics. The benefits of this approach can be found both in military and civilian applications. For military applications, it can help in localisation and tracking of objects of interest. For civilian applications, the similar object localisation procedures can make the criminal investigations more effective, extracting the meaningful data from the massive video footage. Recently, the wide accessibility of consumer unmanned aerial vehicles has become a new threat as even the simplest and cheapest airborne vessels can carry some cargo that means they can be upgraded to a serious weapon. Additionally they can be used for spying that imposes a threat to a private life. The autonomous car driving systems are now impossible without applying machine vision methods. The industrial applications require automatic quality control, including non-destructive methods and particularly methods based on the video analysis. All these applications give a strong evidence in a practical need in machine vision algorithms for object detection, tracking and classification and gave a reason for writing this thesis.

The contributions to knowledge of the thesis consist of two main parts: video tracking and object detection and recognition, unified by the common idea of its applicability to video analytics problems.

The novel algorithms for object detection and tracking, described in this thesis, are unsupervised and have only a small number of parameters. The approach is based on rigid motion segmentation by Bayesian filtering. The Bayesian filter, which was proposed specially for this method and contributes to its novelty, is formulated as a generic approach, and then applied to the video analytics problems. The method is augmented with optional object co-ordinate estimation using plain two-dimensional terrain assumption which gives a basis for the algorithm usage inside larger sensor data fusion models.

The proposed approach for object detection and classification is based on the evolving systems concept and the new Typicality-Eccentricity Data Analytics (TEDA) framework. The methods are capable of solving classical problems of data mining: clustering, classification, and regression. The methods are proposed in a domain-independent way and are capable of addressing shift and drift of the data streams. Examples are given for the clustering and classification of the imagery data.

For all the developed algorithms, the experiments have shown sustainable results on the testing data. The practical applications of the proposed algorithms are carefully examined and tested.

Statement of Originality

I, Dmitry Kangin, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Acknowledgements

The author is pleased to thank Denis Kolev and Mikhail Suvorov, discussions with whom, as well the co-operation on the articles and book chapter, have contributed significantly to this thesis. The discussions, article co-operation and various help and assistance of my supervisors, Professor Plamen Angelov and Professor Garik Markarian, helped me enormously. Also I need to praise Professor George Kolev for all discussions and help. Also, many appreciation go to my colleagues in Rinicom, to fellows of the EU FP7 TRAX project for object tracking, in which I am happy to participate, and to my parents, Nikolay and Lyudmila, and my sister Evgenia.

Contents

List of Figures

List of Tables

Acronyms & Abbreviations

ADALINE – ADAptive LInear NEuron

ARTOT – Autonomous Real-Time Object detection and Tracking

BRISK – Binary Robust Invariant Scalable Keypoints

ECG – electrocardiogram

EKF – Extended Kalman Filter

EM algorithm – Expectation-Maximisation algorithm

GMM – Gaussian Mixture Model

GLOH – Gradient Location and Orientation Histogram

GPS – Global Positioning System

HOG – Histogram of Oriented Gradients

JPDA – Joint Probabilistic Data Association

KKT conditions – Karush-Kuhn-Tucker conditions

MADALINE – Many ADALINE

MCMC – Monte-Carlo Markov chain

MHT – Multiple Hypothesis Tracking

MM algorithm – Majorisation-Minimisation algorithm

MRF – Markov Random Field

MSER – Maximally Stable Extremal Regions

OCR – Optical Character Recognition

OoI – Object(s) of Interest

PHD – Probability Hypothesis Density

QR code – Quick Response code

RDE – Recursive Density Estimation

SIFT – Scale Invariant Feature Transform

SOM – Self-Organising Maps

SURF – Speeded Up Robust Features

SVM – Support Vector Machines

SESAR – Single European Sky Air traffic management Research

TEDA – Typicality-Eccentricity Data Analytics

TLD – Tracking-Learning-Detection

UKF – Unscented Kalman Filter

VIVID PETS 2005 – Video Verification of Identity, Performance Evaluation of Tracking and Surveillance

1 Research Overview

This chapter outlines the research motivation and the summary of the research contributions and publications, as well as the research methodology. The place of the research in the contemporary areas of research is given from both theoretical and practical points of view. The chapter is organised as follows. It starts from the research motivation (section 1.1). After that, the research contribution is described (section 1.2). Then the methodology (section 1.3) and publication summary (section 1.4) are given. The section is finished by the thesis outline (section 1.5).

1.1 Motivation

The machine vision applications are pervasive whilst many scientific and practical challenges still remain unsolved [1]. For example, many of the algorithms still require parameterisation and tweaking for particular practical problems [2]. It invokes a high research interest to this topic and, at the same time, boosts associated areas such as data mining and image processing. In general, the machine vision systems are designated for automatic data extraction and understanding problems for image and video information. These systems can be used on their own or as a part of larger systems, which can also contain algorithms based, for example, on the control systems theory or robotics.

Hereafter only some of the widely known problems, which are now being solved by means of machine vision, are presented. Vehicle detection systems, appearing during the last decades, are based on automatic number plate detection and recognition [3]. Many train logistic systems utilise automatic train number reading [4]. The barcodes and QR codes are detected automatically by terminal devices, such as sales register scanner [5] in the former case and phones [6] in the latter. OCR systems are extensively utilising the methods of symbol recognition [7]. Face detection algorithms are widely used for people appearance detection and identification [8], as well as for documents identification and recognition. Various robotics problems [9], including landscape description [10], automatic object surveillance [11], obstacles detection [12], and many others, need object detection and tracking algorithms. The machine vision algorithms are intensively employed in new automotive applications developments [13]. Such machine vision challenges as road lane mark-up detection, obstacles detection, and collision prevention detection are all being solved by object detection and recognition algorithms [14].

Such evident need in tracking, detection and recognition problem solutions require further development of new efficient algorithms. The practical applications impose various requirements to the algorithm .To define them, it is necessary to explain here several conceptual terms and yet define the restrictions on them, supporting the aim of the work. In this work, the aim is to reduce the amount of the domain-specific constants, pre-defined heuristics, but, at the same time, give some guarantees (analytical or experimental, depends on the problem) that the algorithm will work on various data samples. This property can be referred to as '*universality*'. Additionally, a lot of applications imply that the objects, described by recognition or detection model, change over time, so the recognition and detection algorithms' configurations should be correspondingly changed. The algorithm, capable of augmenting the model without its total re-building, is called '*incremental*'. For many of such algorithms requirements are accompanied with by the restriction of models' parameters number, memory assumption and computational complexity of the algorithm to be independent of the sample set size. Such algorithms can be called *'online'*. For some task, one should impose other restrictions, i.e. require the algorithm to adopt during the change of the model assumptions (number of assigned clusters, number of classes). These algorithms are referred as '*evolving*'.

This research work is focused on object tracking, detection, and recognition problems, as well as it is proposing new algorithms to tackle this problem. It was performed during the part-time PhD course in Lancaster University and combines the results obtained during the PhD research with those obtained from the practical works in the same area, initially in Russia and later in Rinicom, United Kingdom. The work is especially focused on evolving and incremental algorithms and proposes novel algorithms in the field of video tracking and image classification and clustering which can be useful in object tracking and classification problem as well.

1.2 Research Contribution

This research study focuses on the object recognition and detection problems, stressing the problem of incremental learning. During the research, the following main contributions have been achieved:

- two novel methods for object detection and tracking have been developed using the newly proposed Bayesian filtering technique [15], [16], the object detection and tracking techniques were applied and evaluated on the data sets and applied to thermal and optical video data;
- the evolving classifier AutoClass has been developed and evaluated on the image data [17];
- the TEDA framework has been exploited for building up the new classifiers and clustering algorithms [18], [19], [20], [21];

- the Chan-Vese image segmentation algorithm has been improved which lead in significant increase of the algorithm speed, by fitting a Chan-Vese functional to a Boolean programming problem. Applications of the algorithm to medical image analysis were also investigated [22];
- a book chapter has been published jointly with two other co-authors, Denis Kolev and Mikhail Suvorov, describing SVM-based methods, in [23].

1.3 Methodology

The proposed research is focused on object detection and recognition problems. It is comprised of several parts, giving the evidence of the proposed methods:

- theoretical concepts research;
- methods implementation;
- practical application of the concept.

During the theoretical concepts research, the analytical description of the proposed methods has been composed which gives an evidence of the method validity as well as the boundaries of the method application.

Then, the method implementation was designed to show the practical possibility of method usage, as well as to augment the theoretical analysis by practical proof of concept.

The practical implementation gives us an evidence of the method applicability, given the real practical problems to cope with.

1.4 Publication Summary

The research, described in this thesis, was described in the following publications, given in chronological order by the submission date:

1. P. Angelov**,** D. Kangin, X. Zhou, D. Kolev (June 2014), Symbol Recognition with a new Autonomously Evolving Classifier AutoClass, *In Proc. 2014 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS-2014*, 2-4 June, 2014, Linz, Austria
2. D. Kangin, P. Angelov (April 2015) Recursive SVM based on TEDA, The Third International Symposium On Learning and Data Sciences, Egham, UK.
3. D. Kangin, P. Angelov (July 2015) Evolving Clustering, Classification and Regression with TEDA, International Joint Conference on Neural Networks, Killarney, Ireland, 2015.
4. D. Kangin, P. Angelov, J. A. Iglesias, and A. Sanchis (August 2015). "Evolving Classifier TEDAClass for Big Data." Procedia Computer Science 53 (2015): 9-18.

5. D.Kolev, D. Kangin, G.Markarian (July 2015). Data Fusion for Unsupervised Video Object Detection, Tracking and Geo-Positioning, Fusion 2015, Washington DC, USA.
6. D. Kangin, D. Kolev, G.Markarian (October 2015). Multiple Video Object Tracking Using Variational Inference, Sensor Data Fusion: Trends, Solutions, Applications, 10th Workshop, Bonn, Germany.
7. D. Kangin; P. P. Angelov, J. A. Iglesias (2015). Autonomously Evolving Classifier TEDAClass. Journal of Information Sciences
8. D. Kolev, M. Suvorov, and D. Kangin (2016). Kernel models and Support Vector Machines (chapter), P. Angelov (Ed.), *Handbook on Computational Intelligence*, 750pp., World Scientific, ISBN: 978-0-470-28719-4
9. D. Kangin, D. Kolev, P. Angelov (2016, submitted). Fast Non-parametric Image Segmentation Using Majorisation-Minimisation of a Modified Chan-Vese Functional. International Journal of Intelligent Systems.

Also, the following patent application was submitted during the practical works at Rinicom:

1. Rinicom Holdings Limited. Patent GB1415372.0 - Object detection. Lodged 29 August 2014

1.5 Thesis Outline

The remainder of the thesis is organised as follows.

Chapter 2 – Existing tracking, detection and recognition techniques: Contains two parts: tracking algorithms survey, and object detection and pattern recognition for video data survey, with supplementary subpart on feature detection review. The review serves a purpose to reveal the connection between object tracking, detection and pattern recognition problems.

Chapter 3 – Proposed Object Tracking Techniques: Proposes a method for object tracking, as well as introduces combined object detection and tracking method. The method utilises the ideas of rigid motion segmentation, implementing them using the Bayesian filtering technique. The method is presented in two versions: featuring Laplacian and Variational approximations on the update step of the Bayesian filter. The first method was presented at the Fusion 2015 conference [15], while the second one was presented on Sensor Data Fusion [16].

Chapter 4 – Proposed Object Detection and Recognition Techniques: This section describes methods for object detection and recognition. The group of methods, based on TEDA framework, is proposed for different fundamental data mining problems and includes the classification method TEDAClass, the clustering method TEDAClustering, and the regression method TEDAPredict, which were presented in the paper [19], as well as their versions for big data, presented in the paper [21]. They are based on the fuzzy rule structure, proposed in [17]. The exact SVM incremental training algorithm was presented in [18] and described in this chapter, capable of dynamic modification of the kernels and individual slack variables for all vectors from the training data set. Additionally the example of the trainable kernel, which is based on TEDA framework, is given in the same paper. This chapter also contains the description of the new iterative solution for the well-known Chan-Vese functional, featuring MM algorithm, and suggests its non-parametric version.

Chapter 5 – Implementation and validation of the developed algorithms: Contains information on the medical, video surveillance and transportation application of the methods. Also it comprises the information about the experiments which were carried out for these methods.

Chapter 6 – Conclusion and Future Work: summarises the information given in this thesis, as well as defines directions for the further work.

2 Existing tracking, detection and recognition techniques

Many machine vision systems are based on a synergy between several methods, where each of the methods strengthens the overall model. For example, vehicle plate recognition systems combine object detection and recognition methods [24]. Medical systems for blood cells counting may rely on tracking, but detection and recognition are also needed [25], which can be used jointly with tracking [26]. Paying attention to the interaction between these methods in object surveillance models, this chapter gives a review of tracking, detection and pattern recognition methods, giving a theoretical and historical basis for the subsequent research results descriptions in these areas.

2.1 Tracking methods survey

Broadly speaking, tracking problems aim to restore the evolving positions of the objects of interest. For visual object tracking problems (including those based on optical, thermal cameras and radars), which are in the scope of this research, the common interest is caused by the temporal evolution of the object appearance and location on the image.

More thoroughly, suppose a set of sensors. The sensors can be video and thermal cameras, radars, measurements, and also such measurement devices such as GPS sensors, laser range meters and many others, altogether combined into sensor fusion. The data from all these sensors can be used either simultaneously or selectively. One can propose a model of an object of interest (OoI) in terms of sensor data, as well as, possibly, a position of the OoI. Moreover, there can be more than one object, captured by the sensors ("on the scene"), or even there can be no objects. The number of objects on a scene may vary, or be constant, depending of the problem statement.

The visual object tracking methods can be based on different movement models. Obviously, there are no universal models for object tracking (and visual tracking in particular). Instead there are methods (based either on well-grounded theoretical assumptions or on ad hoc suggestions), which can be applied to practical tracking problems. The examples of such methods are 'tracking by detection' [27], Bayesian filter trackers family [28], or their combinations, as well as some methods based on empirical density estimation for object detection and tracking [29].

2.1.1 Brief review of the state-of-the-art tracking methods

This survey begins with general examples of the tracking problem, then it narrows down to particular problems, related to video tracking, and, finally, the algorithms' historical evolution is reviewed.

Nowadays, many lorries, buses and ever consumer vehicles around the UK and all over the world are equipped with GPS vehicle trackers which are used to register a location of each vehicle as well as to see vehicle movement trajectories online. Given data from several satellites and a geographical map, it is possible to tether the actual position of any particular vehicle to geographical objects (i.e. to the closest roads). The planes' GPS trackers help to manage remotely from air traffic offices the occupancy of runways and avoid collisions. Development of such systems is supported by such national and international programmes as SESAR [30] and NextGen [31].

Radars are extensively used for airborne and marine objects detection [32]. Using such systems and imposing model assumptions on characteristics of the OoI, the object can be detected based on the radar data. For military and aviation purposes, radars are extensively used to track the airborne objects, especially for aviation security and foreign planes military invasion detection.

Tracking is extensively applied to the celestial objects [33], featuring analytical non-Bayesian models from time immemorial. For capturing (and, nowadays, filming) of the objects, the telescopes should follow them, hence the models need to take into account movement of the Earth and the objects itself. During the centuries, many sophisticated models were contributed to estimate the model of the objects movement and using it track the objects.

The narrower problem of video tracking has been rapidly emerging during the last decades due to the increase of the contemporary computing platforms' resources. Further in this section the review of the tracking approaches is built as a general description, but bearing in mind video tracking applications.

The description starts with one of the simplest tracking models (which can be used for single or multiple object tracking). Imagine that the initial position of objects in a video feed at the initial instant of time is known (e.g. provided by 'human in the loop'). One can assume that the object is contained within the sufficiently small area surrounding the object (the area size and the similarity relation are strongly dependent on the domain). Then the measurement on the subsequent frame that is the closest to the previous one can be selected as a new position of the object. This approach is straightforward, but it is not reliable. The object may once go out of the area of search; any wrong object registration ruins all further tracking as the movement direction was taken erroneously [34]; also the objects' position is uncertain in the case of absent measurements. The algorithm can be enhanced, however, by some tougher assumptions on the object movement model (e.g. constant speed or acceleration). However, the main problem is that even a single tracking error or absent measurement can cause loss of the object.

Many of the popular object tracking techniques arose from various optimal Bayesian filter approximations. The scope of this review allows to give only brief method descriptions, however the details related to the proposed methods will be explained in detail in chapter 3, where the novel Bayesian tracking models, proposed in this research, are described. The optimal Bayesian filter [35] has the same graphical model as a Hidden Markov model [36] and gives a procedure of estimating the posterior density of the hidden parameters (i.e. object localisation) given the visible sensor data information and dependency of the visible variables on the hidden variables. The optimal Bayesian filter, assuming parametric Gaussian distributions and linear dynamic equations, is referred to as the Kalman filter [37], which has exact analytical solution. Linear Kalman filter [37] can be used for linear movement model that means only steady object movement with constant movement parameters (i.e. velocity vector) and Gaussian noise distribution. Those models which use general non-linear differentiable dynamic equations, can be approximated by the Extended Kalman filter [38], featuring first-order Taylor series approximation to fit it to the model similar to Kalman filter. An alternative way of the approximation which uses the assumed distributions' sufficient statistics is referred to as Unscented Kalman Filter (UKF) [39]. Besides of non-linear Kalman filter approximations, which can help in introducing non-linearity, the model can be composed of several submodels with Markov chain transition between them [41]. Another non-linear model approximations include particle filter, using Monte-Carlo Markov chain (MCMC) approach for distribution estimation [40].

Another problem arises when the methods need to be adopted to consider multiple objects tracking. There are two main concepts which are relative to the problem and inspired the methods proposed in this thesis. The first, rigorous and generic, approach was invented first for radar data and includes classical Bayesian multiple target tracking methods such as Multiple Hypothesis Tracking (MHT) [43], Joint Probabilistic Data Association (JPDA) [44] and Probability Hypothesis Density (PHD) [45], [46] filters.

The second approach is designed especially for video movement detection. It is time-consistent segmentation of all the frames in the video sequence according to the continuous intensity areas, as well as consistent speed characteristics, with following selection of only some of the clusters. This approach is often referred to as 'rigid motion segmentation' [42]. Another productive idea, helping to enhance tracking, is to exploit the measurements retrospective for the objects being tracked. It is featured by many domain-specific video analysis algorithms. One of the most fruitful non-Bayesian methods exploiting this idea, which has arisen in the last decade is Tracking-Learning-Detection (TLD) [26]. This method uses tracking based on optical flows and at the same time trains the classifier for simultaneous detection and tracking of the objects. Then the data is combined in the style of mixture of experts. Such approach extends earlier series of methods, grouped as ‘tracking by detection’, which rely on known appearance of object, make its detection, and compares with the previous frames [27].

Another popular video tracking techniques group include those using non-parametric recursive data density estimation techniques, predominantly not domain-specific and non-parametric. One of the examples of such techniques is Recursive Density Estimation (RDE), applied to object tracking (ARTOT) [47]. This method exploits density estimation using recursively updated Cauchy-type functions. To match the points during the video tracking process, Scale Invariant Feature Transform (SIFT) [48] descriptors are used.

2.1.2 Technical description of the state-of-the-art methods

2.1.2.1 *Bayesian filtering*

In this section the problem of tracking is considered as the hidden states estimation given the visible measurements and dependency model in the scope of the Bayesian filtering framework. In this framework, the model states are modelled by a Markov chain, so that the next state is dependent of the previous state only, and there is a dependency model for the measurements from the hidden states. The visible sensor data are interpreted as the sensory measurements, and the (hidden) states correspond to the parameters of the object being tracked. For example, the state can be the set of actual position of the objects while the measurements are obtained from miscellaneous sensors like camera, compass, GPS sensor, and accelerometer.

The measurements are modelled as dependent only on the current state of the system (first order Markov model). Here we see (hidden) states, $\boldsymbol{x}_k$, and corresponding measurements, also called visible states, $\boldsymbol{z}_k$, where k is the positive discrete time instant. The probabilities $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k)$ are assigned to the transfer between the states, and $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$ are the measurements’ probabilities given the states.

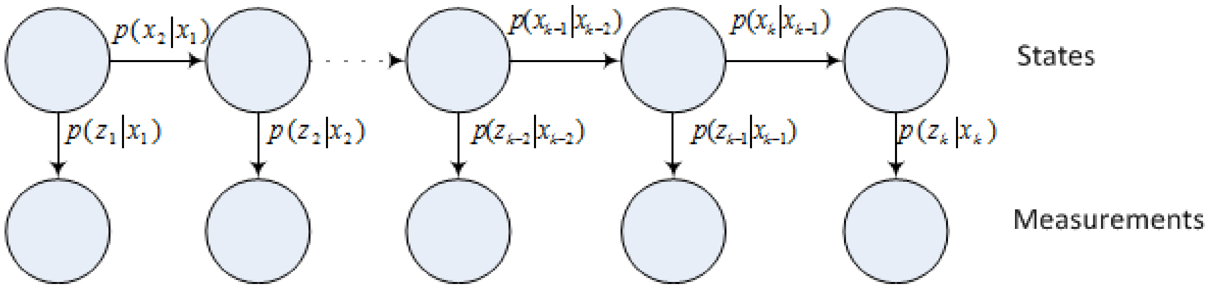


Figure 1 Bayesian filter graphical model

Below is the description of the graphical model depicted in Figure 1. One can see that the graphical scheme is exactly the same as it is for the Hidden Markov Model (HMM) [49], [50].The difference between them is that HMM corresponds to the discrete hidden states only, while Bayesian filter addresses the problem of continuous hidden states estimation. According to the Markov principle, the probability of appearance of the objects given that the previous states are independent of all but the previous state, i.e.

$$p(\boldsymbol{x}_k|\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3,\dots\boldsymbol{z}_{k-1},\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3\dots\boldsymbol{x}_{k-1}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}). \quad (1)$$

Also, it is the model’s property that

$$p(\boldsymbol{z}_k|\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3,\dots\boldsymbol{z}_{k-1},\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_3\dots\boldsymbol{x}_{k-1}) = p(\boldsymbol{z}_k|\boldsymbol{x}_{k-1}). \quad (2)$$

Then

$$p(\boldsymbol{x}_1,\boldsymbol{x}_2,\dots,\boldsymbol{x}_k) = p(\boldsymbol{x}_1)\prod_{i=2}^{k} p(\boldsymbol{x}_i|\boldsymbol{x}_{i-1}), \quad (3)$$

$$p(\boldsymbol{z}_1,\boldsymbol{z}_2,\dots,\boldsymbol{z}_k|\boldsymbol{x}_1,\boldsymbol{x}_2,\dots,\boldsymbol{x}_k) = \prod_{i=1}^{k} p(\boldsymbol{z}_i|\boldsymbol{x}_i). \quad (4)$$

Then denote

$$p(\boldsymbol{x}_{1\dots k}) = p(\boldsymbol{x}_1,\boldsymbol{x}_2,\dots,\boldsymbol{x}_k), \quad (5)$$

$$p(\boldsymbol{z}_{1\dots k}|\boldsymbol{x}_{1\dots k}) = p(\boldsymbol{z}_1,\boldsymbol{z}_2,\dots,\boldsymbol{z}_k|\boldsymbol{x}_1,\boldsymbol{x}_2,\dots,\boldsymbol{x}_k). \quad (6)$$

After Bayes theorem application one can see

$$p(\boldsymbol{x}_{1\dots k}|\boldsymbol{z}_{1\dots k}) = \frac{p(\boldsymbol{z}_{1\dots k}|\boldsymbol{x}_{1\dots k})p(\boldsymbol{x}_{1\dots k})}{p(\boldsymbol{z}_{1\dots k})} \propto p(\boldsymbol{z}_{1\dots k}|\boldsymbol{x}_{1\dots k})p(\boldsymbol{x}_{1\dots k}). \quad (7)$$

Here $p(\boldsymbol{z}_{1\dots k}) = \int_{\boldsymbol{x}_{1\dots k}} p(\boldsymbol{z}_{1\dots k}|\boldsymbol{x}_{1\dots k})p(\boldsymbol{x}_{1\dots k})d\boldsymbol{x}_1 d\boldsymbol{x}_2 \dots d\boldsymbol{x}_k$.

As one can see, hidden variables are integrated out from the denominator; hence the denominator can be treated as a constant with respect to the hidden variables and, therefore, can be excluded from the problem solution.

The parameters learning problem can be stated as a complete-data log-likelihood function optimisation problem:

$$\ln p(\boldsymbol{x}_{1\ldots k}, \boldsymbol{z}_{1\ldots k}|\Theta) = \ln p(\boldsymbol{x}_1|\Theta) + \sum_{i=2}^{k} \ln p(\boldsymbol{x}_i|\boldsymbol{x}_{i-1}, \Theta) + \sum_{i=1}^{k} p(\boldsymbol{z}_i|\boldsymbol{x}_i) \to \max_{\Theta}\ , \tag{8}$$

where Θ denotes the parameters of the system. Alternatively, parameters can be determined by the model itself in the way, depending of the aim of the modelling.

One of the prominent particular models of Bayesian filter family, which features a closed form solution, is the Kalman filter [37]. It assumes particular parameterisation of the Bayesian filter for the linear system. Consider that the hidden states x_k are distributed as $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = \mathcal{N}(\boldsymbol{x}_k|A_k\boldsymbol{x}_{k-1}, B_{k-1})$, $p(\boldsymbol{z}_k|\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{z}_k|C_k\boldsymbol{x}_k, D_k), k > 1,$ where $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$ is the normal distribution of the hidden variables $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance matrix Σ, $p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1|\boldsymbol{\mu}_1, \Sigma_1)$, and $A, B, C, D, \boldsymbol{\mu}_1, \Sigma_1$ are the model parameters. This model is indeed a version of the Bayesian filter, and it can be proven [35] that

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1..k-1}) = \mathcal{N}\left(\boldsymbol{x}_k\middle|\tilde{\boldsymbol{\mu}}_k, \tilde{\Sigma}_{\mathrm{k}}\right), \tag{9}$$

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1..k}) = \mathcal{N}(\boldsymbol{x}_k|\boldsymbol{\mu}_k, \Sigma_{\mathrm{k}}), \tag{10}$$

$$p(\boldsymbol{z}_k|\boldsymbol{z}_{1..k-1}) = \mathcal{N}(z_k|C_k\tilde{\boldsymbol{\mu}}_k, \Sigma_{\mathrm{k}}), \tag{11}$$

denoting

$$\tilde{\boldsymbol{\mu}}_k = A_k\boldsymbol{\mu}_{k-1} \tag{12}$$

$$\tilde{\Sigma}_k = A_k\Sigma_{k-1}A_k^T \ (\textit{prediction step}), \tag{13}$$

and then, using notation from (12) and (13)

$$\boldsymbol{v}_k = \boldsymbol{z}_k - C_k\tilde{\boldsymbol{\mu}}_k, \tag{14}$$

$$S_k = C_k\tilde{\Sigma}_kC_k^T + D_k, \tag{15}$$

$$K_k = B_k\tilde{\Sigma}_kS_k^{-1}, \tag{16}$$

$$\boldsymbol{\mu}_k = \tilde{\boldsymbol{\mu}}_k + K_k\boldsymbol{v}_k, \tag{17}$$

$$\Sigma_k = \tilde{\Sigma}_k - K_kS_k\,K_k^T \ (\textit{update step}). \tag{18}$$

The full proof of these equations is widely known and given in many tutorials like [35]. The undoubted advantage of the model is its simplicity and exact solution. However, one of the largest restrictions of the Kalman filter model is an assumption of the model linearity. There are different ways to relax this assumption.

To introduce the non-linear model extensions, it is convenient to represent Kalman filter model [35] as

$$\boldsymbol{x}_k = A_k\boldsymbol{x}_{k-1} + \boldsymbol{\varepsilon}_k, \tag{19}$$

$$\boldsymbol{z}_k = C_k\boldsymbol{x}_k + \boldsymbol{\xi}_k, \tag{20}$$

where k is an instant of time, $\boldsymbol{\xi}_k, \boldsymbol{\varepsilon}_k$ is a normally distributed noise, $\boldsymbol{x}_k$ are hidden variables at the time k, A_k, B_k, C_k are the coefficients dependent of time.

The Extended Kalman filter (EKF) [38] gives the solution for the non-linear problem

$$\boldsymbol{x}_k = g(\boldsymbol{x}_{k-1}) + \boldsymbol{\varepsilon}_k (\text{state equation}), \tag{21}$$

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{\xi}_k (\text{measurement equation}). \tag{22}$$

Here one can see that the Kalman filter addresses a particular case of the extended Kalman filter model. The EKF problem is solved by linearisation of the filtering equations, thus making them analogous to the Kalman filter. The solution uses first-order Taylor series approximation around the state expectation $\boldsymbol{\mu}_{k-1}$:

$$g(\boldsymbol{x}_{k-1}) \approx g\ (\boldsymbol{\mu}_{k-1}) + A_k(\boldsymbol{x}_{k-1} - \boldsymbol{\mu}_{k-1}), \tag{23}$$

$$A_k = \nabla f(\boldsymbol{\mu}_{k-1}), \tag{24}$$

where ∇ is a vector differential operator (nabla operator).

For the measurement equation, the similar approximation is applied:

$$h(\boldsymbol{x}_k) \approx h(\tilde{\boldsymbol{\mu}}_k) + C_k(\boldsymbol{x}_k - \tilde{\boldsymbol{\mu}}_k), \tag{25}$$

where

$$C_k = \nabla h(\tilde{\boldsymbol{\mu}}_k). \tag{26}$$

In all other aspects, the EKF exploits the general results for Kalman filter as it is described in formulae (9)-(18); $\tilde{\boldsymbol{\mu}}_k$ is also understood in sense of the previous Kalman filter description.

Another way of linearisation is provided by the Unscented Kalman filter (UKF), featuring the unscented transform. It deterministically builds a set of so-called ‘sigma points’, which give sufficient statistics for the corresponding Gaussian distribution. The sigma points p^* for normal distribution $\boldsymbol{x} = \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ are defined [39] as

$$\boldsymbol{p}^0 = \boldsymbol{\mu}, \tag{27}$$

$$\boldsymbol{p}^{i\pm} = \boldsymbol{\mu} \pm \left(\sqrt{(n+\lambda)\Sigma}\right)_i, i = 1..n. \tag{28}$$

Here n is the dimensionality of the vector space for the normal distribution, $\lambda = \alpha^2(n+\beta) - n$, and α, β are the model parameters. Then the random distribution is transformed via generally non-linear function $\boldsymbol{x}_k = g(\ \boldsymbol{x}_{k-1})$.

$$\boldsymbol{\mu}_k = w_\mu^0 g(\boldsymbol{p}^{0,}) + \sum_{i=1}^{n} w_\mu^{i+} g(\boldsymbol{p}^{i+}) + \sum_{i=1}^{n} w_\mu^{i-} g(\boldsymbol{p}^{i-}), \tag{29}$$

$$\Sigma_k = w_c^0 C_k^0 + \sum_{i=1}^{n} w_c^{i+} C_k^{i+} + \sum_{i=1}^{n} w_c^{i-} C_k^{i+}, \tag{30}$$

$$C_k^* = (g(\boldsymbol{p}^{*,}) - \boldsymbol{\mu}_k)(g(\boldsymbol{p}^{*,}) - \boldsymbol{\mu}_k)^T, \quad (31)$$

$$w_\mu^0 = \frac{\lambda}{n+\lambda}, w_c^0 = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta), \quad (32)$$

$$w_c^{i\pm} = w_\mu^{i\pm} = \frac{1}{2(n+\lambda)}, i = 1 \ldots n. \quad (33)$$

Here $\boldsymbol{p}^{i-}$ are obtained from $\boldsymbol{\mu}_{k-1}$ and Σ_{k-1}.

The full derivation is out of the scope of this thesis chapter, however it can be found in miscellaneous sources like [39].

2.1.2.2 *Multiple object tracking filters*

The problem formulation, described in the previous section, is quite general, nevertheless it does not address many practical problems [51] such as:

- multiple objects tracking (at least if data association mechanism is not defined)
- object tracking with clutter, i.e. measurements which are not relevant to the object appearance
- object tracking with manoeuvring (at least neither it allows to switch between models nor provides a framework for dynamic evolution of a single model)

Therefore, there are two ways to take these problems into account:

a) reject Bayesian filter model and replace it with an alternative one, in the most radical case, propose ad-hoc solution for the particular problem;
b) add the missing parts of the model or generalise it in such a way that takes into account the complications of the problem as well.

Non-Bayesian filtering techniques can be based on such well-known algorithms as TLD [26], which features optical flow tracking jointly by classifier trained for previous appearances of the same object. The problem of data association can be solved by the classifier: the classification rediscovers the object based on their appearance. The algorithm can be implemented as separate tracking procedures for each of the objects, but in order to rationally utilise the computational resources, the feature extraction stage needs to be shared between all the objects. However, the algorithm does not provide an initial detection model. This means that it is still needed to provide a separate algorithm for object detection. Another problem is that the TLD algorithm is domain specific, it means that it is defined for video object tracking only. Another possible approach is to use non-parametric density estimation methods, described in the next section.

The description of the Bayesian filtering techniques starts from the case of multiple target tracking, when the initial tracking positions are known. Then it is possible to create a single-target Bayesian filter, i.e. Kalman filter, for each of the OoIs, using the measurements which are the closest to it (the OoI). This approach is widely used, but it has noticeable drawbacks. First, the approach needs modifications if there is any possibility of missing measurements (i.e. if the objects' measurements are not present at least in one of the stages), otherwise it will lose the track on the first measurements loss. Second, if there is a possibility of close interaction between the targets, there is no mechanism to prevent target switching, as it does not take into account existence of other targets. In other words, if the erroneous measurements are assigned or if the measurements are absent, it affects all subsequent tracking unless the approach is combined with data association algorithms.

Therefore, there is a strong need in multi-target data association algorithms which are capable of assigning the measurements to the targets and determining that the measurements are missing for the current objects at certain stage. Most known models of such kind are Multiple Hypothesis Tracker (MHT) [43] and Joint Probabilistic Data Association (JPDA) [44]. Where additionally there is no initial position known and there is clutter, there is also an evident need in the detection model as well. For manoeuvring objects tracking, i.e. when the objects' model changes in time, the selection of the most appropriate model needs to be provided. As an example, it can be Markov chain of trackers fitted to some particular sub-model.

MHT data association model [43] generates a set of hypotheses about associations between measurements and targets including those associations in the past stages, with corresponding probability of correctness for each of the hypotheses. The final posterior distribution is built as a mixture of distributions for each of the hypotheses with weights, showing hypotheses' probabilities. The problem with this method is the exponential growth of the hypotheses quantity, as the retrospective association is taken into account. To cope with it, there is a need to (heuristically) prune the hypotheses with small probability.

JPDA data association model [44] does not reassess the past associations; instead, it recursively updates the previous hypothesis, assigning probability of measurement association with each of the targets. Therefore, the method does not need to maintain a set of hypotheses, but, instead, it updates a single hypothesis, derived from the previous stages.

Additionally, there is a family of pointillist filters, and, in particular, widely renowned PHD filter [45]. In contrast to MHT or JPDA associations, these filters do not explicitly provide neither target tracks nor target associations, but, the detection mechanism is built into the model. Instead, they compute the intensity map over target space, which provides density of the targets in each point of the space. The intensity maps integral over all the space is finite and gives the estimated number of the targets. However, they can also be coupled with methods providing target tracks associations, which is beyond the scope of this thesis.

2.1.2.3 *Non-parametric density estimation techniques for tracking*

As an alternative to Bayesian tracking, there are non-parametric density estimation techniques for tracking, e.g. described in [47] and developed further in [52].

There exists widely renowned density estimation method, which is based on the kernel trick. This technique is referred to as Kernel Density Estimation (KDE) [53] and can be applied to either univariate or multivariate distributions and gives the following estimation of the probability distribution $p(\boldsymbol{x})$ based on data:

$$p(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} K(\boldsymbol{x}, \boldsymbol{x}_i), \tag{34}$$

where $\boldsymbol{x}$ is the approximation point, and $\boldsymbol{x}_i, i = 1 \dots N,$ is a set of data samples from the probability distribution under approximation, $K(\boldsymbol{x}) \geq 0, \int K(\boldsymbol{x}) = 1.$ The last condition is the normalisation condition.

The kernel function can be of various kinds. The most well-known functions are Parzen window kernel

$$K_L(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \frac{1}{h^D}, & \frac{\|\boldsymbol{x} - \boldsymbol{y}\|}{h} \leq \frac{1}{2}, \\ 0, & \text{else,} \end{cases} \tag{35}$$

where h is the window size parameter, or Gaussian window kernel

$$K_G(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2h^2}\right), \tag{36}$$

where h is the standard deviation parameter.

The ARTOT method, described in [47] and [54], aims to avoid the parameters dependency, such as the h parameter, as well as recursive incremental description. It estimates data density value according to the equation

$$D(x) = \frac{1}{1 + \|\boldsymbol{x} - \boldsymbol{\mu}_k\|^2 + \mathrm{X}_k - \|\boldsymbol{\mu}_k\|^2}. \tag{37}$$

Here, the training data sequence $\{\boldsymbol{x}_1, \dots \boldsymbol{x}_k\}$ is considered, $\boldsymbol{\mu}_k$ is a mean value for the data set of k samples, and Σ_k is the mean of the squared data samples vectors.

The mean of data and squared data are updated according to the following equations:

$$\boldsymbol{\mu}_1 = \boldsymbol{x}_1, \boldsymbol{\mu}_k = \frac{(k-1)}{k}\boldsymbol{\mu}_{k-1} + \frac{1}{k}\boldsymbol{x}_k, \tag{38}$$

$$\mathrm{X}_1 = \|\boldsymbol{x}_1\|^2, \mathrm{X}_k = \frac{(k-1)}{k}\mathrm{X}_{k-1} + \frac{1}{k}\|\boldsymbol{x}_k\|^2, \tag{39}$$

where $(\mathrm{X}_k - \|\boldsymbol{\mu}_k\|^2)$ can be interpreted as a variance of the data norm.

In ARTOT method [47], [54], the density is calculated for each of the pixels, also the mean and variance of the density are calculated:

$$\text{mean: } \overline{D}_k(\boldsymbol{x}) = \frac{k-1}{k}\overline{D}_{k-1}(\boldsymbol{x}) + \frac{1}{k}D_k(\boldsymbol{x}), \overline{D}_1(\boldsymbol{x}) = D_1(\boldsymbol{x}), \tag{40}$$

$$\text{mean of the squared value:} \tag{41}$$

$$\overline{D}_k^2(\boldsymbol{x}) = \frac{k-1}{k}\overline{D}_{k-1}^2(\boldsymbol{x}) + \frac{1}{k}[D_k(\boldsymbol{x})]^2, \overline{D}_1(\boldsymbol{x}) = [D_1(\boldsymbol{x})]^2,$$

$$\text{variance: } \left[\bar{\sigma}_{D_k}(\boldsymbol{x})\right]^2 = \overline{D}_k^2(\boldsymbol{x}) - \left(\overline{D}_k(\boldsymbol{x})\right)^2. \tag{42}$$

Then, the event of sudden change of the density more than on $\bar{\sigma}_{D_k}(\boldsymbol{x})$ is considered. If it occurs, then it is recognised as a moving object presence event. It constitutes the detection stage in the algorithm .The tracking in this method is made by matching Scale Invariant Feature Transform (SIFT) descriptors of the points of interest [48].

2.1.3 Optical flow: the necessary supplement to video object tracking

The optical flow concept was proposed by the psychologist James J. Gibson in 1950 amongst other contributions on video perception [60]**.** The subsequent scientific works on optical flow affected not only theoretical psychology studies, but also computer vision applications. The video tracking often requires to define a correspondence between the same points, depicted on different frames. To estimate the motion of the whole object, we need to consider the movement of its individual points.

Optical flow methods estimate the video frame points' velocities in video frame co-ordinate system. These methods are divided into two groups:

i) dense, and

ii) sparse.

The methods from the first group calculate the velocity map for all points within the image, while those from the second group estimate velocities for some pre-specified set of points only, where the number of elements in this set is much less than the total amount of points on the image. Usually, the points are selected using the criteria of their conformity with the background estimation models.

One of the widely renowned dense optical flow estimation methods is Horn-Schunck method [55]. It is based on the optimisation problem for the energy functional defined for the image $I(x, y)$ as

$$E = \int\int \left(\frac{\partial I(x,y)}{dx} u(x,y) + \frac{\partial I(x,y)}{dy} v(x,y) + \frac{\partial I(x,y)}{dt} \right)^2 \mathrm{d}x\, \mathrm{d}y + \\ + \int\int \kappa^2 (\|\nabla u(x,y)\|^2 + \|\nabla v(x,y)\|^2)\, \mathrm{d}x\, \mathrm{d}y \to \min_{u,v}, \tag{43}$$

where κ is the regularisation constant, and $\kappa^2(\|\nabla u(x,y)\|^2 + \|\nabla v(x,y)\|^2)$ is a Tikhonov regularisation term. The equations are solved using iterative scheme, which is described in [55].

Lucas-Kanade method [56] is an example of sparse optical flow approach. In this method, the following system of linear equations is considered for each of the points:

$$AV = B, \tag{44}$$

$$A = \begin{bmatrix} \frac{\partial I(x_1,y_1)}{\partial x} & \frac{\partial I(x_1,y_1)}{\partial x} \\ \cdots & \cdots \\ \frac{\partial I(x_n,y_n)}{\partial x} & \frac{\partial I(x_n,y_n)}{\partial x} \end{bmatrix}, V = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, B = \begin{bmatrix} -\frac{\partial I(x_1,y_1)}{\partial t} \\ \cdots \\ \frac{\partial I(x_n,y_n)}{\partial t} \end{bmatrix}. \tag{45}$$

In this case, the points $x_1 \ldots x_n$ are the points in the vicinity of the point of interest, and we are aiming to find the vector V. Obviously, as the linear equations system has more equations than variables, it is proposed to apply the least squares analytical optimisation method instead.

To increase the stability of the algorithm, such techniques are exploited using pyramidal subsampling [57]. In this method, the pyramid is built for the image, where the upper (0-th) layer is a source image in its original resolution, and each of the lower layers is built as a subsampling of the previous one. Then, starting from the lowest level, it is possible to estimate the optical flow and then enhance the estimation on each of the stages.

2.2 Detection and recognition methods survey

Pattern recognition problems aim to assign the labels to the objects or their parts, given sensor measurements corresponding to them. This set of problems are divided on different branches, from which clustering, classification and regression are in the scope of this research.

To illustrate the difference between these problems, a few examples can be given. First, given images of digits (0 ... 9), one can assign digit label to each of the images. Such kind of the problems, where the labels are pre-defined, are often described in terms of supervised learning, i.e. classification or regression, where the classifier is adjusted, or trained, on some given training data set. A very peculiar case of classification problem is one class classification, dubbed as anomaly or outlier detection, where there is a prior knowledge only on one of the object classes, usually predominant in the training set. Then the model assesses conformity with these classes and considers all the remaining elements as a clutter. It is the assumption of this approach that the model will be general enough to conform to the recognition data set, which is, generally speaking, different from training data set.

Semi-supervised techniques, such as reinforcement learning, rely on partly available labelling. It means that the labelling is available for small percent of the data or the labelling 'oracle' can say 'correct' or 'not correct' only after the algorithm execution for the input vector. The motivation of these models can be to avoid extensive training set generation for detection quality improvement.

Another problem appears when the data from the sensor are not labelled at all (e.g. because of the prohibitively large training data set). And even more, the labels may not be defined at all, and the problem is to separate it into reasonable (w.r.t. some criteria) subsets. In this case, it can be solved by a clustering problem. For example, it can be an image grouping according to some pre-defined similarity and within some pre-defined feature space, aimed to group images of similar objects. These approaches can complement each other: clustering may be used as a preliminary stage before the classification.

Object detection is a particular application of pattern recognition techniques, which aims to distinguish the data relevant to the OoI from the sensor data. The problem is often solved by means of classification between the OoI appearances and all other measurements, which are referred as clutter. For example, vehicle brand logo detection problem can be stated as a classification between the logo and non-logo images. More generally, the application of pattern recognition for pictorial information is described in section 2.2.1, one of the particularly noticeable example of object detection algorithms, pattern matching, is discussed in section 2.2.8.

Pattern recognition history is closely connected with the computer systems evolution. First propositions and implementations of automatic pattern recognition were proposed in 1950s, along with the rise of neurophysiologist studies [58]. In these studies, the aim was not to present some algorithm or framework, capable of solving particular or general recognition problems, as it is widely considered in contemporary computer science, but to model brain operation in principle as a part of human brain operation studies. This area, firstly developed by neurophysiologists, has given a simultaneous rise to a large class of recognition models – neural networks, which were introduced into emerging computer science. As a result, the 1970-1980s were marked with intensive development of neural networks for pattern recognition. These networks, which are discussed further in section 2.2.2, gained both theoretical and practical popularity, because they provided a universal framework for pattern recognition problems, predominantly classification.

Another part of studies was devoted to decision trees, which are discussed in section 2.2.3. These studies were aimed to state the decision based on some sequence of rules, given by a tree walk, where each of the nodes performs elementary data mining operation.

Another popular data mining framework, Support Vector Machines (SVM), was founded by Vapnik, Lerner and Chervonenkis in 1960s [96], [97], but was undeservedly forgotten until the beginning of 1990s. The initial method was proposed for binary classification problem, but there are versions for clustering, regression and multiclass classification. The method considers the binary classification as between-class margin maximisation, which can be represented as a quadratic programming problem. This group of methods, based on SVM, is described in section 2.2.4.

Another branch, fuzzy logic, arose from fuzzy set theory proposal by Lotfi Zadeh in 1965 [59]. The branch has initially emerged as an extension of the Boolean algebra, where the binary answer ‘true’ or ‘false’ is replaced by the normalised quantity of belonging to one of two classes. This branch, further described in section 2.2.5, has an astonishingly large range of practical applications, including standard pattern recognition problems, but also automatic control theory and robotics.

Data clustering is an important group of techniques amongst the overall pattern recognition branch. In the scope of this work, particularly important techniques are k-means and deeply connected with them Gaussian mixture models together with the EM algorithm, used for maximum likelihood optimisation for Gaussian mixtures. These techniques, together with a brief review of spectral clustering, are presented in section 2.2.6.

From data clustering problem one particular practical problem has split, related to image analysis: image segmentation. The image segmentation problem is discussed in different ways in the literature, and considering the scope of the proposed algorithms in the thesis, a few of contemporary segmentation methods are observed in section 2.2.7.

One of the biggest challenges for pattern recognition methods, including those observed before, is feature extraction, described in section 2.2.9. For some algorithms, it can be incorporated to the algorithm (like in convolutional neural networks), and for other ones, it is built independently. Some of them are general, other ones are suitable for some particular application, which can be often effective especially in the case of lack of resources (like embedded platforms).

2.2.1 Object detection methods review

For several decades object detection has been remaining an active research area. The approaches for this problem can be divided by several ways. Some of them are domain-specific, i.e. they rely on the particular models' assumptions, some of them are general, based on the common methods of machine learning and independent of the particular practical problem. This section is aimed to show the brief retrospective of video object detection techniques and at the same time show the contemporary place of pictorial object detection in a scope of overall pattern recognition.

Object detection can be divided into two groups of methods: supervised (referred as segmentation) and unsupervised. On the other side, the methods can be data driven, i.e. the model is a function of some data set, or can be expert-driven, where the opinion of the experts is expressed in some (mathematical) model. The choice of the model should be justified by particular practical problem properties and a number of factors, such as computational resources, memory consumption, adequacy of the model to the practical problem, and many others.

The geometrical methods for machine object detection and scene understanding appeared in 1960s. Lawrence G. Roberts developed in his PhD thesis [185] solid polyhedrons recognition methods, based on vectorisation of the image containing solid primitive. Sobel-Feldman [62] operator, proposed in 1968, and Prewitt [63] operator, proposed in 1970, provided simple, but very productive ideas of object contrast edges detection which lead to substantial progress in geometrical image understanding. In parallel, various template matching techniques were developed. One of the well-known works in this direction was part-based model, proposed by Fischler and Elschlanger in 1973 [186]. In this method, the object was recognised by separate pattern matching of the parts of the image rather than all the pattern. The developments based on pattern matching and geometrical image understanding, along with general machine learning techniques developments, as neural networks, described in section 2.2.2, or MacQueen's k-means algorithm [61], described in section 2.2.6.1, determined the object detection techniques for quarter century. The area of interest has changed dramatically in the last two decades, when various image descriptors have appeared, such as Scale Invariant Feature Transform (SIFT) [48], Speeded Up Robust Features (SURF) [64], Binary Robust Invariant Scalable Keypoints *(*BRISK) [65], region descriptor Maximally Stable Extremal Regions (MSER) [66], model-based feature descriptors [67], and trainable feature descriptors [68], which are all reviewed in section 2.2.9. Now they are widely combined with well-known machine learning techniques such as boosting trees, SVMs and neural networks, described in sections 2.2.2, 2.2.3, 2.2.4, respectively.

2.2.2 Neural networks review

The neural networks ideas arose from brain studies, which attracted researchers on from time immemorial. In the middle of the 20th century, whilst the electrical circuits were growing more compound, the ideas of creating artificial neural networks, modelling the natural ones and aimed to resolve perception and recognition problems, similar to those faced by human beings, appeared. One of the first well known mathematical models of the neurons, implemented in electrical circuits, was Threshold Logic Unit, proposed by Warren McCulloch and Walter Pitts in 1943 [69]. This neuron employed Heaviside step function model, working as a threshold. But the real interest to the neural networks theory was awakened by Donald Hebb's book 'The Organization of Behavior' [70]. One of the pioneering neural networks models, using McCulloch-Pitts neuron models was perceptron, proposed by Frank Rosenblatt [71].

Here the perceptron algorithm is described in order to outline its relation to Support Vector Machines, described in section 4.3.2. The perceptron algorithm was first proposed by Frank Rosenblatt [71] in 1962, and now it is used for various classification problems. This neural network consisted of the linear model accompanied with the non-linear threshold function responsible for the model output:

$$f(\boldsymbol{x}) = \text{sign}\,[\boldsymbol{w}^T \varphi(\boldsymbol{x})], \tag{46}$$

where $\boldsymbol{x}$ is the input vector, $\varphi(\boldsymbol{x})$ is feature transformation, $\boldsymbol{w}$ are the weights. The sign of the function determines the objects' class (let $f(\boldsymbol{x}) = 1$ be class A and $f(\boldsymbol{x}) = -1$ be class B). To train the classifier, one need to state the optimisation problem, minimising the classification error for the model. One can see that the maximisation of sum of correctly classified objects cannot be carried out using gradient descent because this sum is not continuous but piecewise constant function of $\boldsymbol{w}$. On the other side, for each of the patterns, the misclassification error for the pattern can be given as $\max(0, \boldsymbol{w}^T \phi(\boldsymbol{x}_i) y_i)$, where $\{x_i\}, i = \{1 \dots n\}$, is the training set and $y_i \in \{-1, 1\}, i = \{1 \dots n\}$, is the correct class label. One can see that the error will be positive if and only if the input vector is classified correctly. The corresponding minimisation problem is

$$E(w) = -\sum_{i=1}^{n} \max(0, \boldsymbol{w}^T \phi(\boldsymbol{x}_i) y_i) \rightarrow \min_{\mathbf{w}} . \tag{47}$$

It is still non-differentiable and cannot be optimised analytically, but at least stochastic gradient descent can be applied. After then, various model extensions, called as multilayer perceptron [72], were proposed to allow multiple class classification as well as defining much wider, non-linear classes of interclass boundaries using a chain of perceptrons.

One of the pioneering artificial neural networks, designed for the practical application, referred as ADALINE/MADALINE and based on the Least Mean Squares algorithm [73], was proposed by Bernard Widrow and Marcian Hoff in Standford University in 1962 [74].

However, these networks were created in condition of extremely limited hardware resources, which did not allow to build the models adequate to complex practical problems. In 1970s and 1980s the situation with the computational resources improved enough for new, more complicated neural networks. Based on the developing neuroscience achievements [75], Self-Organising Maps (SOM) were proposed by Teuvo Kohonen [76]. Another significant contribution, shedding a light on associative memory, was John Hopfield's Hopfield network [79]. Convolutional networks began to appear at this time, beginning from the works by Kunihiko Fukushima [77]. Later they were significantly improved by Yann LeCun and other researchers [78]. These networks are characterised by composite structure composed of many layers, often with non-homogeneous layers (i.e. each of the layers is designed for particular function, e.g. feature extraction or smoothing). The training is often made layer-wise, i.e. there is no training for all the network at the same time but for separate layers only which hindered global optimisation of the network structure.

Nowadays, there are many neural networks designed for object detection and recognition. Seemed to be out of attention for some time, the convolutional networks began to develop extensively during the last decade, enabling new opportunities of recognition between hundreds of object classes [80]. While the first models, developed in 1980s, suffered from intensive manual parameterisation (which, however, can be replaced by cross-validation that gives better statistical grounds for parameterisation but barely more tractable), state-of-the-art networks provide better generalisation whilst using parameters learnt from data.

Spiking neural networks theory is also developing at this time. The ideology behind this group of models is to increase the realism of the models of natural neural networks by artificial ones. These networks are not prepared yet for practical applications but are used more for neurophysiological studies. However, extensive research is being carried out by different contributors [81], [82], including NeuCube model by Nikola Kasabov, [83], and Steve Furber's SpiNNaker project in Manchester University [84].

On the border between fuzzy systems and neural networks, the neuro-fuzzy system branch has emerged [85], [86], [87]. It incorporates evolving systems which are capable of 'adopting' to the pattern change during the time in the data sequences, based on both the ideas of fuzzy rule systems and neural networks.

Nowadays, despite the rise of other pattern recognition methods, described in this review, neural networks, notably simple perceptron-based models and complex convolutional networks, are still popular and serve various purposes, from primitive well-separable data classification to pattern classification problems defined for hundreds or (sometimes) even thousands of classes [88]. They serve to a wide range of practical models, ranging from ECG analysis [89] to robotics [90].

2.2.3 Decision trees

Decision trees are classification and regression algorithms, mapping the data from the input vectors in the feature data space to the output labels, or, possibly, regression values, using the multi-stage partitioning scheme. For this purpose, the system is organised into the tree, each node of which represents some elementary classification or regression algorithm. The output values are mapped to the leaves.

The most evident idea behind the decision trees is to map some expert knowledge to the form of the tree, which was used from time immemorial (see Figure 2).

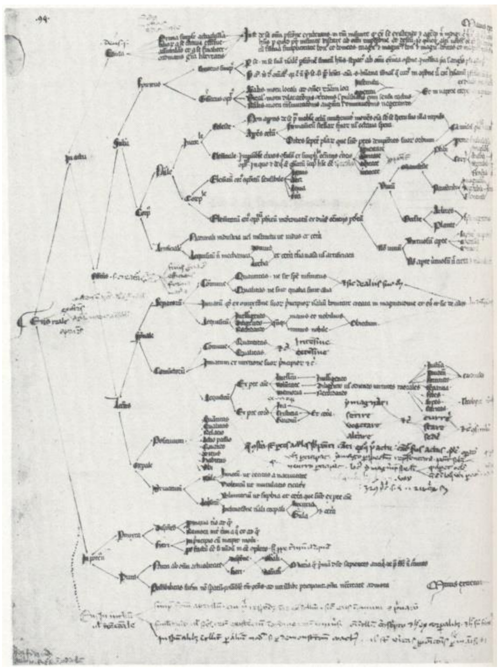


Figure 2 Classification of the states of 'being', *Electorium magnum*, by Thomas le Myésier, around 1323 (borrowed via [187] from [188]).

Probably the most straightforward strategy is to compose a tree-like structure for dividing the data space by thresholds to provide piecewise-linear dividing surface, e.g. ensured by expert knowledge. For example, in Figure 3 one can see both two-dimensional space representation (where vector $\boldsymbol{x}$ from this space is represented by its co-ordinate projections (x^1, x^2)) and the corresponding decision tree model.

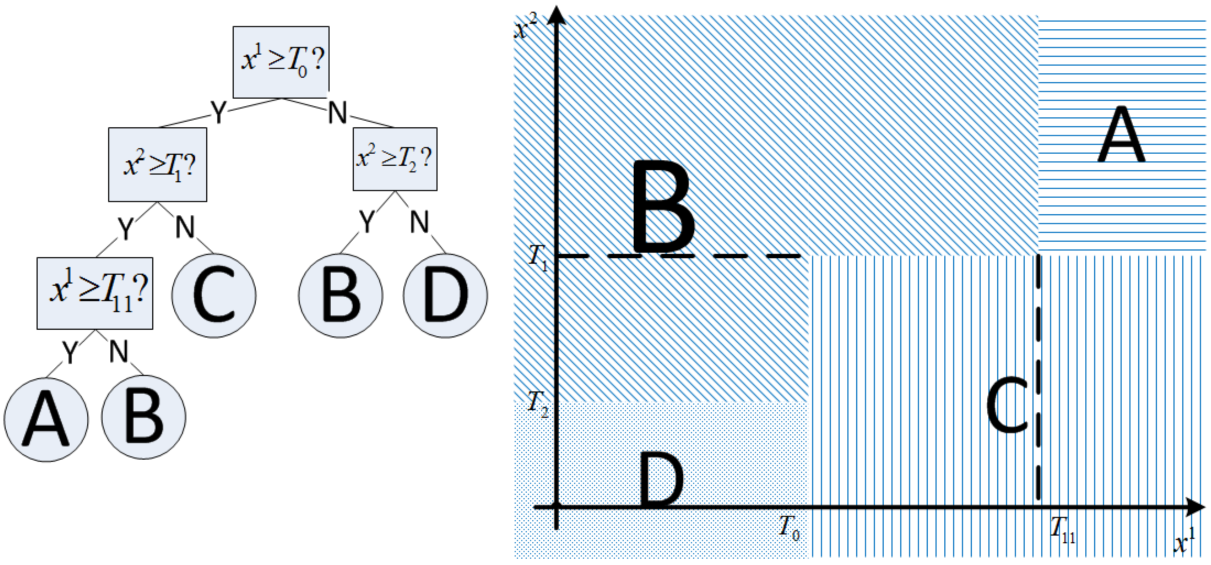


Figure 3 Example of the decision tree with the corresponding feature space

However, models, based on the thresholds, are highly restrictive as they assume a finite set of the data types and limited choice of barriers, whilst the real data may vary tremendously. Also, as it is expert driven, it is ordained to be subjective. Hence, when applying to classification problems, expert-driven decision trees are an appropriate solution only for some trivial problems with unknown 'ground truth' dividing surfaces deemed to be simple.

As an alternative, one can consider more complex classifiers on each of the stages with non-linear borders, but the question arise about the universal automatic procedure of decision tree building. As it was with many other systems, since the appearance of computers the keenest idea was to automatise the tree building considering as a tree of independent models. The following ideas of automation have arisen during the last decades:

- *committee methods* is a large group of methods, based on the idea of unification of the answers from several independent models [91], e.g. by averaging the responses from the models;
- *(threshold-based) classification and regression trees* [92] is an approach for automatic building (and, what is also important, pruning) of the threshold-based trees like depicted in figure 3;
- *boosting* [93] proposes the structure, based on chain of models, where the next model

enforces the previous one in order to tackle misclassification.

Boosting methods are being successfully used in machine learning applications because they provide a regular way to combine many hundreds or even thousands of classifiers on each of the stages. One of the prominent examples of this approach, Viola-Jones algorithm, was proposed in 2001 [8] for face recognition, using many primitive Haar classifiers [94] with AdaBoost boosting algorithm [95].

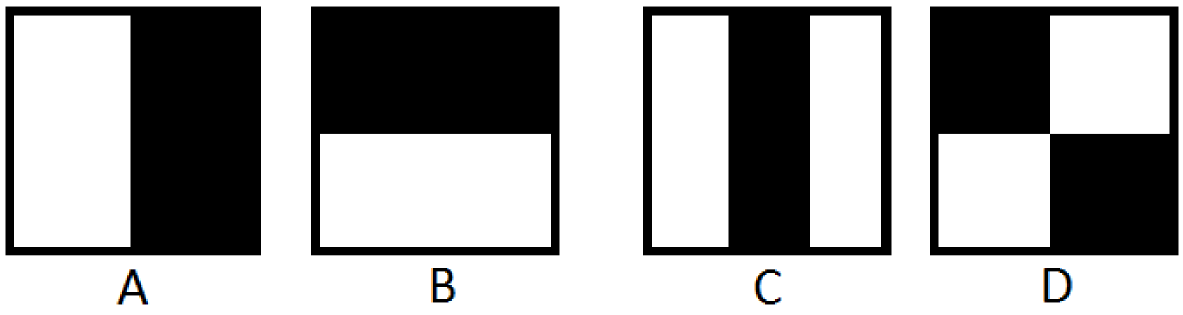


Figure 4. Haar-like features graphical representation.

The idea of feature extraction, exploited by this algorithm, is astonishingly simple and nevertheless is proven to be effective for the practical applications. In this algorithm, different sub-windows of the fixed size (e.g. 24×24) are selected. For each of these windows, Haar-like features are selected. The features divide the pixels into two groups, and then subtract one group from the other. Different partitions are acceptable. In the original Viola-Jones algorithm, the data is partitioned on left and right half-window, top and bottom, on central and peripheral parts, or diagonally (Figure 4):

$$f_i\big(I(x,y)\big) = \int I(x,y)D_i(x,y)\mathrm{d}x, \tag{48}$$

where $I(x,y)$ is an two-dimensional image, $f_i\big(I(x,y)\big)$ is the i-th feature extraction function, and $D_i(x,y)$ is a discriminantative function of the data, $D_i(x,y) \in \{-1, 1\}$, where the value of $D_i(x,y)$ corresponds to the colour in Figure 4.

If one takes all possible sliding windows of the given size with a set of Haar-like features, the dimensionality of such feature space and the complexity of the classifier training is prohibitively huge. Hence some preliminary feature selection method is needed. The selection of features together with the decision tree building can be made by the boosting algorithm, like AdaBoost [95].

2.2.4 Support Vector Machines

Support Vector Machines (SVM) appearance is dated back to 1963, when Generalised Portrait Method [96] was proposed by Vladimir Vapnik and Alexander Lerner. After then, several papers appeared during almost thirty years of Vladimir Vapnik's co-operation with Alexey Chervonenkis [97], but the method was undeservedly overlooked for decades. The genuine popularity came along with the SVM algorithm formulation featuring kernel trick [98]. Since this article the algorithm has become enormously popular both with and without kernel trick, resulting in formulation of various extensions, including those for regression [99], structured learning [100], multi-class classification [101], and even clustering [102]. Due to the kernel trick, kernel engineering contributed significantly to the popularity method, [103]. Below the method's description is given.

First, consider a two class classification problem, where k is a number of data samples, $\Omega_L = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k\}$ is a training data set, $\Upsilon = \{-1, 1\}$ is the class labels set, there exists some labelling function $\Psi(\boldsymbol{x}_n) = t_n, n \in [1 \dots k]$. Contrary to the perceptron classifier, described in section 2.2.2, which is used for the same problem formulation, here the linear separating hyperplane is selected using margin maximisation instead of the error minimisation criterion for perceptron. First, consider that the data set is linear separable, it means that there exists a hyperplane which exactly divides points of the given classes. The separating hyperplane criterion is functionally formulated as follows: select the hyperplane which has the largest distance from both the closest training set elements. One can prove that this hyperplane will be equidistant from both data sets [200]. All these considerations can be summarised in the following margin optimisation problem [200]:

$$\frac{1}{\|\boldsymbol{w}\|} \min_n [t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b)] \to \max_{\boldsymbol{w},b} . \tag{49}$$

Here $y(x) = \boldsymbol{w}^T \boldsymbol{x} + b,$ and $\boldsymbol{w}, b$ are the hyperplane parameters, and $\phi(\boldsymbol{x}_n)$ is a feature mapping from the data set space to the features vector space. However, this problem formulation is still difficult for optimisation. One can mention that the coefficients $\boldsymbol{w}$ can be rescaled. Then one can declare

$$\min_n [t_n(\boldsymbol{w}^T \phi(\boldsymbol{x}_n) + b)] = 1 \tag{50}$$

that corresponds to the points closest to the separating hyperplane.

Then the optimisation problem can be formulated as

$$\frac{1}{\|\boldsymbol{w}\|} \to \max_{\boldsymbol{w},b} , \tag{51}$$

$$t_n(\boldsymbol{w}^T \phi(x_n) + b) \geq 1. \tag{52}$$

Finally one can notice that it is equivalent to the quadratic programming problem [200]

$$\frac{1}{2} \|\boldsymbol{w}\|^2 \to \max_{\boldsymbol{w},b} , \tag{53}$$

$$t_n(\boldsymbol{w}^T\phi(x_n)+b)\geq 1. \tag{54}$$

Then, it is necessary to generalise this formulation to non-separable classes, e.g. there exists no hyperplane that exactly divides two classes. For this purpose special slack variables are introduced that have non-zero values when the data points are out of their classes' boundaries. The version of the problem statement for non-separable, or overlapping, classes, referred as C-SVM, can be stated as follows:

$$\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{n=1}^{k}\Xi_n\to\min_{w,b,\Xi_n}, \tag{55}$$

$$t_n y(\boldsymbol{x}_n)\geq 1-\Xi_n, \Xi_n\geq 0, n=1\dots k. \tag{56}$$

Here Ξ_n is a slack variable for the data point $\boldsymbol{x}_n$, C is a parameter, usually referred as the 'box constraint' [104]. Note that, contrary to the formulation for perceptron, mentioned in section 2.2.2, the analytical optimisation of the problem is possible as it is a quadratic programming problem for both formulations of the problem (53),(54) and (55), (56).

Then one can convert the problem (55), (56) to the dual one, which allows using the kernel trick. For this purpose, the Lagrangian $\check{L}(\alpha)$ is to be constructed:

$$\check{L}(\alpha)=-\frac{1}{2}\sum_{n=1}^{k}\sum_{m=1}^{k}\alpha_n\alpha_m t_n t_m\langle\phi(\boldsymbol{x}_n),\phi(\boldsymbol{x}_m)\rangle+\sum_{n=1}^{k}\alpha_n\to\min_{\alpha,b}, \tag{57}$$

$$0\leq\alpha_n\leq C, \sum_{n=1}^{k}\alpha_n t_n=0. \tag{58}$$

Here $\{\alpha_n\}$ are the Lagrange multipliers.

Denote $\langle\phi(\boldsymbol{x}_n),\phi(\boldsymbol{x}_m)\rangle$ as $k(\boldsymbol{x}_n,\boldsymbol{x}_m)$. $k(\boldsymbol{x}_n,\boldsymbol{x}_m)$ is referred as a positive definite kernel if it is symmetric and positive semidefinite [195]. Then, according to Mercer theorem [201], $k(\cdot,\cdot)$ can be decomposed as $k(\boldsymbol{x}_n,\boldsymbol{x}_m)=\langle\phi(\boldsymbol{x}_n),\phi(\boldsymbol{x}_m)\rangle$ if and only if it is a kernel.

After then, the kernel table, where each row and column corresponds to the element of the training set, can be calculated for non-zero elements of the sum in the Lagrangian (they will be margin or erroneously classified training set vectors). This representation has an advantage that the kernels can be defined for infinite dimensional functional spaces and hence it is possible to calculate kernels even for the cases where the analytical representation of the feature mapping $\phi(\cdot)$ is impossible.

Another version of the problem is ν-SVM [105]:

$$\frac{1}{2}\|\boldsymbol{w}\|^2-\nu\gamma+\frac{1}{k}\sum_{n=1}^{k}\Xi_n\to\min_{w,b,\gamma,\Xi_n}, \tag{59}$$

$$t_n y(\boldsymbol{x}_n) \geq \gamma - \Xi_n, \Xi_n \geq 0, \qquad n = 1, \dots, k. \tag{60}$$

It can be proven that the parameter ν gives the lower bound of the support vectors' fraction and at the same time the upper bound for the margin vectors [105]. If $\gamma > 0$ and $\hat{C} = \frac{1}{k\gamma}$, the problem can be proven to be equivalent to C-SVM problem[105].

The algorithm cannot be applied directly to multiclass problems, however there are a plenty of technical methods to make it feasible. The SVM classifiers can be arranged into the network exploiting the principles 'one versus all', 'one versus one', or into directed acyclic graphs, which is actually a decision tree allowing to classify object using exactly $(K-1)$ binary SVM classifiers [106] (Figure 5).

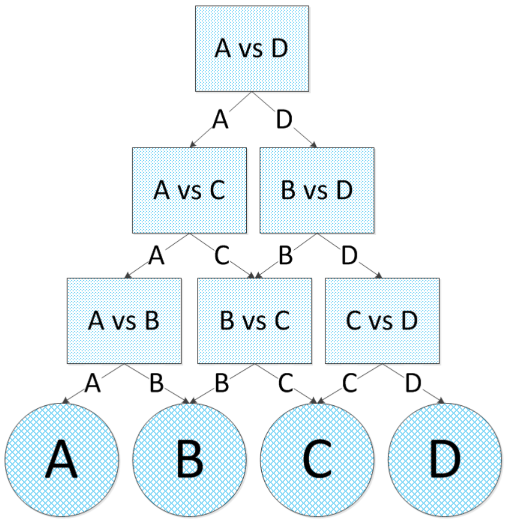


Figure 5 Direct Acyclic graph for multiclass SVM using multiple SVM classifiers [106]

The one-class classification version of the SVM classifier was proposed in [107] for anomaly detection. While the problem statement modification is not dramatic and is also a particular case of quadratic programming, it allows using the method for broader class of practical tasks. It is designed to make the best separation margin between normal and anomalous data using the hypersphere. This problem can be solved using the modified formulation of ν-SVM which separates between outlier and inlier data with maximal margin by the hypersphere in the feature space:

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N} \xi_i - \rho \to \min_{\boldsymbol{w},\xi,\rho}, \tag{61}$$

$$\langle w, \phi(x_i)\rangle \geq \rho - \xi_i, \xi_i \geq 0, i = 1 \dots N, \tag{62}$$

where the parameters are expressed the same way as for ν-SVM.

2.2.5 Evolving fuzzy classifiers

The evolving fuzzy rule based systems were initially proposed by Lotfi Zadeh in 1965 [59]. Since then, the theory of fuzzy sets and fuzzy systems was developed. The fuzzy systems are a set of fuzzy rules, to each of which each of the input vectors partially belong with some weights between 0 and 1. Several types of rules were proposed by L. Zadeh [108] and E. Mamdani and S.Assilian [109] (Zadeh-Mamdani fuzzy rule system), then by T.Takagi and T. Sugeno in [110] (Takagi-Sugeno fuzzy rule system):

Zadeh-Mamdani:

$$IF\ \left(\mathrm{ant}_i(\boldsymbol{x})\right)\ THEN\ (y_i\ IS\ Y_i), \tag{63}$$

Takagi-Sugeno:

$$IF\ (\mathrm{ant}_i(\boldsymbol{x}))\ THEN\ (y_i = x^T\Theta_i). \tag{64}$$

Here y_i is an outcome of the i-th rule, $i \in [1..N]$, Θ_i is a design matrix for linear regression, $\boldsymbol{x}$ is an input data vector, and $\mathrm{ant}_i(\boldsymbol{x})$ is the antecedent of the fuzzy rule. The antecedent is expressed the same way for both the rules as

$$\mathrm{ant}_i(\boldsymbol{x}): x^1 is\ L_i^1\ AND\ x^2 is\ L_i^2\ AND\ x^n is\ L_n^i, \tag{65}$$

where n is a dimensionality of the vector $\boldsymbol{x} = (x^1, x^2, \dots x^n)$, and $\boldsymbol{L}^i = (L_1^i, L_2^i \dots L_n^i)$ is a reference vector for the fuzzy rule.

Recently, AnYa-type of the fuzzy rules was proposed [111], which provides generalisation of Zadeh-Mamdani and Takagi-Sugeno systems' antecedents:

$$\text{AnYa: } IF\ (\boldsymbol{x} \text{ is like } \boldsymbol{L}_i)\ THEN\ (y_i = \boldsymbol{x}^T\Theta_i), \tag{66}$$

where $'\mathrm{like}'$ is a predicate for belonging of the vector $\boldsymbol{x}$ to the fuzzy rule's context $\boldsymbol{L}_i$. Here one should not treat multi-dimensional data component by component, but create a single predicate defined on vectors [111].

Initially, fuzzy rule systems were designed for expert-based systems making it just a more flexible alternative to decision trees due to weights of belonging. It seriously restricted applications and a range of the problems by certain automatic control problems, which could be solved with the fuzzy approach, until, exactly the same way as it happened with decision trees, effective idea of learning automation had made all the difference. With automatic fuzzy rules generation, it is possible to build the fuzzy systems with no preliminary knowledge about the data or with some very restricted, fixed amount of parameters known beforehand. In this context, Evolving Fuzzy Systems approach has emerged [112]. This context addresses the following problems to build machine learning systems [113], where first three are folded into each other:

- *Incremental learning*: the training algorithm does not require to re-train the model on the old data, when the additional new data is to be used for training;
- *Online design*: the system accumulates in the memory only a limited context extracted from the data, not full data sets;
- *Evolving structure*: the system can be adopted 'from scratch', without any pre-set structure, by incremental and decremental (i.e. deletion of some of the parts of the model) learning, enforcing steady patterns and rejecting anomalous ones or those which have not been discovered for a long time;
- *Speed and memory efficiency*: the algorithm should be computationally effective enough for real-time data stream processing.

Formulation of the evolving systems paradigm resulted in the development of many algorithms [112]. One example of the evolving fuzzy classifiers is *eClass* [114]. This classifier has been described in two versions. First of them is *eClass0*, featuring Mamdani-type rules (63), and second one is *eClass1,* based on Takagi-Sugeno rules (64).

Define a data sequence $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_k\}$ from the vector space $\mathbb{R}^m$ as well as its finite set of labels Y, assigned to each of the elements of the data sequence. The classifier training procedure for both of the methods is based on the introduction of the potential. The potentials are built upon a RDE method [115] for density estimation and has two varieties. First of them is global density

$$P_k(\boldsymbol{x}_k) = \frac{1}{1 + \frac{\sum_{i=1}^{k-1} d(\boldsymbol{x}_k, \boldsymbol{x}_i)}{k-1}}, \tag{67}$$

where $d(\boldsymbol{x}_k, \boldsymbol{x}_i)$ is a distance or similarity between $\boldsymbol{x}_k$ and $\boldsymbol{x}_l$, ordinarily, but not obligatory, Euclidean or cosine. The second one is local density

$$P_k^l(\boldsymbol{x}_k) = \frac{1}{1 + \frac{\sum_{i=1}^{Q_k^l-1} d(\boldsymbol{x}_k, \boldsymbol{x}_i)}{Q_k^l - 1}}, \tag{68}$$

where Q_k^l is a support of, or the number of elements belonging to, the class l. It can be calculated recursively as it is described in [114]. i-th fuzzy rule is described by its focal point $\boldsymbol{x}_k^{i,*}$. In *eClass0* new fuzzy rule is created, if

$$P_k^l(\boldsymbol{x}_k) > P_k^l\left(\boldsymbol{x}_k^{i,*}\right) \; \forall \; i = L_1(l) \dots L_{N(l)}(l), \tag{69}$$

where $L_i(y_k)$ are the indices of the fuzzy rules of the class l, to which $\boldsymbol{x}_k$ belongs, $N(y_k)$ is the number of the fuzzy rules created for the class y_k.

In *eClass1* the new fuzzy rule is created, if

$$P_k(\boldsymbol{x}_k) > P_k\left(\boldsymbol{x}_k^{i,*}\right) \,\forall\, i = 1 \dots N, \tag{70}$$

where N is the total number of the fuzzy rules, $N(y_k)$ is the number of the fuzzy rules created for the class y_k. One can note that here the global density is used instead of local, per-class density.

In *eClass0* the outputs for some $\boldsymbol{x}$ with unknown class label are derived from the 'winner takes all' rule:

$$y(\boldsymbol{x}) = \arg\max_{i=1\dots N} \tau_i(\boldsymbol{x}), \tag{71}$$

where τ_i define a fuzzy rule firing level. τ_i is defined as

$$\tau_i(\boldsymbol{x}) = \prod_{j=1}^{n} \exp\left(-\frac{1}{2}\left(\frac{d\left(\boldsymbol{x}_k^{i,*}, \boldsymbol{x}\right)}{r_{ij}}\right)^2\right), i = [1 \dots N], j = 1 \dots m. \tag{72}$$

Here r_{ij} is a standard deviation from the focal point for each of the fuzzy rules, which can be exactly calculated recursively during the learning stage [114].

For *eClass1*, the outputs of each fuzzy rule $y_i(\boldsymbol{x})$ are normalised to obtain

$$\bar{y}_i(\boldsymbol{x}) = \frac{y_i(\boldsymbol{x})}{\sum_{j=1}^{N} y_j(\boldsymbol{x})}, \tag{73}$$

where $y_i(\boldsymbol{x}) = \boldsymbol{x}\Theta_i, i = 1 \dots N,$

$$\Theta_i = \begin{bmatrix} \Theta_{i,01} & \Theta_{i,02} & \cdots & \Theta_{i,0K} \\ \Theta_{i,11} & \Theta_{i,12} & \cdots & \Theta_{i,1K} \\ \vdots & \vdots & \ddots & \vdots \\ \Theta_{i,m1} & \Theta_{i,m2} & \cdots & \Theta_{i,mK} \end{bmatrix} \tag{74}$$

is adjusted via fuzzily-weighted RLS algorithm [116]. Here K is a number of the classes.

The output of *eClass1* is given as

$$y(\boldsymbol{x}) = \sum_{i=1}^{N} \frac{\tau_i \bar{y}_i(\boldsymbol{x})}{\sum_{j=1}^{N} \tau_j}. \tag{75}$$

The label of the class is determined as $\arg\max_{i=1\dots m} [y(\boldsymbol{x})]_i$.

Many varieties of this algorithm exist, such as DEC [117], [118], [119] and many others. The difference between them is mainly in the new cluster creation criteria, but some of them also rely on different techniques of clusters inspection, carried out to exclude old clusters, which are not active for a long time, from the model.

2.2.6 Clustering techniques

2.2.6.1 *K-means clustering, Mixture of Gaussians and EM algorithm*

In this section, two connected clustering techniques are described. First of them, k-means clustering, was proposed by MacQueen in 1967 [61]. The second, EM algorithm, was formulated independently for applications going far beyond from the clustering problem. However, there is a strong connection between the EM algorithm for Gaussian mixtures, which is widely used for clustering, and k-means, which can be interpreted as generalisation [120].

The description starts with k-means algorithm. Define a data set $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \dots \boldsymbol{z}_N\}$. For each of the data samples, the cluster labels are assigned from the finite set $Y = \{y_1, y_2 \dots y_M\}$. Also the indicator variable γ_{ik} is introduced, $i = [1 \dots N], k = [1 \dots M]$:

$$\begin{cases} \gamma_{ik} = 1, \text{if } \boldsymbol{z}_i \text{ is from the cluster labelled as } y_k, \\ \gamma_{ik} = 0, \text{else.} \end{cases} \tag{76}$$

For each of the clusters y_k a prototype $\boldsymbol{\mu}_k$ is defined. The objective function $f(Z, Y)$ is declared the following way, leading to the following optimisation problem:

$$f(Z, Y) = \sum_{i=1}^{N} \sum_{k=1}^{M} [\gamma_{ik} \rho(\boldsymbol{z}_i, \boldsymbol{\mu}_k)] \to \min_{b_{ik}, \boldsymbol{\mu}_k}, \tag{77}$$

where $\rho(\boldsymbol{z}_i, \boldsymbol{\mu}_k)$ is a distance between the sample and the prototype (typically Euclidean). One can check that the optimisation in k-means cannot be performed by straightforward analytical differentiation, hence it is needed to provide an alternative optimisation procedure.

This procedure is described as follows. Initially some arbitrary values for $\boldsymbol{\mu}_k$ are fixed that enables analytical optimisation of $f(Z, Y)$ with respect to z_{ik}. After then the weights γ_{ik} are fixed, and $\boldsymbol{\mu}_k$ is optimised analytically with respect to the previous values of γ_{ik}. The algorithm is repeated until the convergence, given the previous stage values of $\boldsymbol{\mu}_k$ to find new values of γ_{ik}. Using simple analytical optimisation technique, one can get

$$\gamma_{ik} = \begin{cases} 1, k = \arg\min_{\mathrm{m}} \rho(\boldsymbol{z}_i, \boldsymbol{\mu}_m), \\ 0, \text{otherwise.} \end{cases} \tag{78}$$

Then (for the Euclidean case)

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} \gamma_{ik} \boldsymbol{z}_i}{\sum_{i=1}^{N} \gamma_{ik}}. \tag{79}$$

This sequence of iterative optimisation constitutes k-means algorithm.

It is the property of the k-means algorithm that it does not provide any weights of belonging for each of the clusters. In many applications it is critical to have this information as well, for example to improve an accuracy of the overall model which depends on the clustering.

For this purpose, it is possible to provide soft clustering that means assignment of weights to each of the elements of the data set for each cluster rather than assigning each element to exactly one cluster. It can be implemented using Gaussian mixture distribution [120] written as

$$p(\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_k, \Sigma_k)\,, 0 \le \pi_k \le 1, \sum_{k=1}^{K} \pi_k = 1, \tag{80}$$

where π_k are the weights of the normal distribution, and $\boldsymbol{\mu}_k, \Sigma_k$ are correspondingly the mean and the covariance of the normal distributions.

The problem of the Gaussian mixture clustering can be represented as the approximation of the empirical distribution of the given data $Z = \{\boldsymbol{z}_1, \dots \boldsymbol{z}_n\}$. by the mixture of the certain number of Gaussians. For this purpose, one can state the maximum likelihood problem for the mixture of Gaussians:

$$p(Z|\pi_{1\dots k}, \boldsymbol{\mu}_{1\dots k}, \Sigma_{1\dots k}) = \prod_{i=1}^{N} \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k, \Sigma_k) \right) \rightarrow \max_{\boldsymbol{\mu}, \pi, \Sigma}\,, \tag{81}$$

$$\sum_{k=1}^{K} \pi_k = 1. \tag{82}$$

Unfortunately, as with k-means, the problem cannot be solved in one step by analytical optimisation. Instead one can fix the posterior probability of the cluster k given the data sample $\boldsymbol{z}_n$:

$$\gamma_{nk} = \frac{\pi_k \mathcal{N}(\boldsymbol{z}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{m=1}^{K} \pi_m \mathcal{N}(\boldsymbol{z}_n|\boldsymbol{\mu}_m, \Sigma_m)}, \tag{83}$$

and then take the likelihood logarithm (it does not change extremum because it is a monotonous function) and differentiate it with respect to the parameters $\pi_{1\dots k}, \boldsymbol{\mu}_{1\dots k}, \Sigma_{1\dots k}$, substituting γ_{nk}:

$$\boldsymbol{\mu}_k = \frac{1}{\sum_{n=1}^{N} \gamma_{nk}} \sum_{n=1}^{N} [\gamma_{nk} \boldsymbol{z}_n]\,, i = 1 \dots k, \tag{84}$$

$$\Sigma_k = \frac{1}{\sum_{n=1}^{N} \gamma_{nk}} \sum_{n=1}^{N} [\gamma_{nk} (\boldsymbol{z}_n - \boldsymbol{\mu}_k)(\boldsymbol{z}_n - \boldsymbol{\mu}_k)^T]\,, i = 1 \dots k, \tag{85}$$

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma_{nk}}{N}. \tag{86}$$

As a summary, the process of the optimisation is given in two iterative steps. First of them, the E-step, is calculation of the posteriors γ_{nk} according to the formula (83). Second of them, the M-step, is calculation of the parameters $(\boldsymbol{\mu}_k, \Sigma_k, \pi_k)$ according to (84)-(86).

One can see that this approach is rather close to the one described for k-means algorithm: the formula (78) of the k-means corresponds to the E-step, and the formula (79) corresponds to the M-step of the EM algorithm.

However, the correctness of such substitution remains unclear and from the description the algorithm does not look like a general approach to the likelihood optimisation. In order to come to the general approach and show the sketch of the correctness proof, the alternative interpretation of the EM-algorithm for Gaussian mixtures is considered below, followed by the generalised EM description. Now denote $X = \{\boldsymbol{x}_1, \dots \boldsymbol{x}_N\}, \boldsymbol{x}_i = \{x_{i1}, x_{i2}, \dots x_{iK}\}$, where $x_{ik} = 1$, if the point belongs to the cluster, and $x_{ik} = 0$ else. Consider the following complete log-likelihood expectation optimisation problem [120]:

$$\mathbb{E}_Z \ln[p(X, Z|\boldsymbol{\mu}, \Sigma, \pi)] \to \max_{\mu,\pi,\Sigma} , \tag{87}$$

$$\sum_{k=1}^{K} \pi_k = 1. \tag{88}$$

After taking the logarithm one can obtain

$$\mathbb{E}_Z[\ln p(X, Z|\boldsymbol{\mu}, \Sigma, \pi)] = \mathbb{E}_Z \ln \prod_{i=1}^{N} p(\boldsymbol{z}_i|x_i)p(x_i) = \tag{89}$$

$$= \mathbb{E}_Z \ln \prod_{i=1}^{N} \prod_{k=1}^{K} \pi_k [\mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k, \Sigma_k)]^{x_{ik}} =$$

$$= \mathbb{E}_Z \left[\sum_{i=1}^{N} \sum_{k=1}^{K} x_{ik} (\ln \pi_k + \ln \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k, \Sigma_k)) \right].$$

Finally, the expectation of the logarithm is expressed as

$$E_Z[\ln p(X, Z|\boldsymbol{\mu}, \Sigma, \pi)] = \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik} (\ln \pi_k + \ln \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k, \Sigma_k)) \to \max_{\boldsymbol{\mu},\pi,\Sigma} \tag{90}$$

After that analytical optimisation for the same expressions (83)-(86) should be carried out as discussed before. It means that the optimisation procedure, given by these formulae, is actually the complete log-likelihood optimisation expectation, repeated in turn with the re-estimation of the posterior distribution.

The general problem, addressed by the EM algorithm, is formulated for the continuous variables case as follows [120]:

$$p(Z|\Pi) = \int_X p(X, Z|\Pi) dX \to \max_{X,\Pi}, \tag{91}$$

where Π are the parameters, Z are visible and X are hidden variables. The discrete formulation is the same except the summation signs replace integration. The assumption is that the optimisation of the original likelihood is difficult, but the complete data likelihood optimisation is much simpler. Then the following decomposition is used:

$$\ln p(Z|\Pi) = \mathfrak{L}\left(p(X), \Pi\right) + KL\left(p(X) \| p(X|\mathrm{Z}, \Pi)\right). \tag{92}$$

Hereafter $\mathfrak{L}\left(p(Z), \Pi\right)$ is referred as a lower bound (further description shows, why it is referred by this name) and is defined as

$$\mathfrak{L}\left(p(X), \Pi\right) = \int\limits_X p(X) \ln\left\{\frac{p(X, Z|\Pi)}{p(X)}\right\} dX\,; \tag{93}$$

$$KL\left(p(X) \| p(X|\mathrm{Z}, \Pi)\right) = -\int\limits_X p(X) \ln\left\{\frac{p(X|\mathrm{Z}, \Pi)}{p(X)}\right\} dX \tag{94}$$

is the Kullback-Leibler divergence [202].

EM algorithm is interpreted in terms of this decomposition as follows. First, on the E-step, the lower bound is maximised with fixed parameters (i.e. $\mathfrak{L}\left(p(X), \Pi^{\mathrm{old}}\right)$) with respect to $p(X)$. One can prove that Kullback-Leibler divergence is non-negative, with zero if $p(X|\mathrm{Z}, \Pi) \equiv p(X)$ (that explains the name ‘lower bound’) [120]. That means, given that $\ln p(Z|\Pi)$ does not depend on the hidden variables X, equivalence of the lower bound maximisation and Kullback-Leibler divergence minimisation.

Second, on the M-step, the lower bound is maximised with respect to the parameters given fixed hidden variables distribution $p(X)$. It increases the lower bound and at the same time changes Kullback-Leibler divergence, which after then does not conform to the minimum conditions with respect to $p(X)$. After then, the new E-step is carried out. The procedure repeats until convergence (e.g. of the lower bound values or to the parameter values). To follow more thorough derivation, see [120].

One can see that this interpretation reveals that the EM algorithm is a particular case of the MM algorithm [166], because on each stage the approximated function is a minorisation of the original function, exactly equal at the start point on the iteration. The convergence properties of the EM algorithm in the general case are beyond the scope of this description and can be found in [189].

The general EM algorithm is widely used in the proposed tracking algorithms described in section 3. In this section, the thorough derivation of the EM algorithm as a part of the proposed tracking model will be given.

2.2.6.2 *Spectral clustering*

Another popular group of clustering methods is spectral clustering [121]. This set of methods exploits similarity matrix eigenvalues analysis for the data partitioning. In this section, as a reference example, Shi-Malik algorithm [121] is briefly observed, which was formulated in 2000 for images, but can be generalised for any particular data vectors. However, the optimisation problem, discussed below, is only one of many problem statements, which are used in the spectral clustering algorithm family. Some alternative spectral clustering models are reviewed in [122]. One can see that any image I can be represented as a full weighted undirected graph $G = (V, E)$, where V the graph vertices are, and E are the edges. Each of the vertices from the set V can be labelled with some features which are contained within the image for the pixels, typically intensity. The edges E reflect neighbourhood relation between pixels. The aim is to find such a partition of the vertices set $V_1 \cup V_2 \dots \cup V_N = V$ that gives higher similarity within the vertex subset, and lower similarity between the subsets.

Consider the problem of partitioning of the data set into two subsets, A and B. Also, for each of the vertices v_i one can assign a label $x_i \in \{-1, 1\}$. Denote the vertices as $V = \{v_1, \dots v_N\}$, and the weights of the edges between v_i and v_j as $w(i, j)$. Then the weights sum from some particular vertex v_i to all other vertices can be denoted as $d(i) = \sum_{1 \le j \le N} w(i, j)$. Then, according to Shi-Malik algorithm, the following optimisation problem is stated [121]:

$$-\frac{\sum_{x_i>0, x_j<0} w(i,j) x_i x_j}{\sum_{x_i>0} d(i)} - \frac{\sum_{x_i<0, x_j>0} w(i,j) x_i x_j}{\sum_{x_i<0} d(i)} \to \min_x, \tag{95}$$

which represents a normalised cut cost.

Then, denote matrix D as

$$D = \begin{bmatrix} d(1) & 0 & \dots & 0 \\ 0 & d(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d(N) \end{bmatrix} \tag{96}$$

and matrix W as

$$W = \begin{bmatrix} w(1,1) & w(1,2) & \dots & w(1,N) \\ w(2,1) & w(2,2) & \dots & w(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ w(N,1) & w(N,2) & \dots & w(N,N) \end{bmatrix} \tag{97}$$

This optimisation problem solution can be approximated by the second eigenvector of the following generalised eigenvector equation:

$$(D - W)\boldsymbol{y} = \lambda D\boldsymbol{y}, \tag{98}$$

where $\boldsymbol{y}$ is an indication vector for data partition. In Shi-Malik algorithm, it is proven that the solution is given by the second smallest eigenvalue [121]. This method is proposed for binary clustering, but the partitioning can be repeated recursively, then the method will be capable of processing as many clusters as needed.

2.2.7 Image segmentation techniques

Image segmentation can be considered as a particular case of a more complex data mining problem, clustering, discussed in section 2.2.6. It is used for localisation of the image parts, corresponding to different areas in the image, which can represent objects of interest. However, as the segmentation results are obtained in unsupervised fashion, image segmentation does not state explicitly the appearance of the object, but gives only some rational decomposition (w.r.t. the model) of the image on several parts. These parts can be labelled (by another algorithm) a posteriori either as objects or as clutter. Together these operations constitute object detection problem solution that justifies the relation of the problem to this research. In this section, some specific groups of models for image segmentation are discussed related to the thesis scope.

A few examples of the image segmentation applications can be given. For blood cells detection it seems rational to group the pixels by spatial neighbourhood and intensity, and then label them with respect to the area size. For text detection, segmentation algorithm can provide connected areas for each of the letter based on the intensity difference between the letter and the background (the symbols with several connected components, which are common in many languages, as á, ú, ы, need post-processing then). After then these components can be labelled as parts of the text or clutter using some features, e.g. area, horizontal or vertical dimensions. In this research, general purpose segmentation algorithms are considered. It means that the (unsupervised) image segmentation is separated from domain specific post-processing stages.

Many algorithms take into account the conditions of intensity homogeneity. Apparently, the simplest segmentation algorithm to deal with (not necessary continuous) homogeneous intensity areas extraction is thresholding. The segmentation results can be specified for the grey scale image I as:

$$\psi(x, y) = \begin{cases} 1, & I(x, y) \geq \theta, \\ 0, & I(x, y) < \theta, \end{cases} \tag{99}$$

where θ is a threshold parameter, calculated according to some heuristics or by solving the optimisation problem like in the Otsu method [124]. In this thresholding method, the hypothesis is that the single threshold can be used globally image.

However, in some cases it is impossible because of diversity of the image. Then there is a need to use of an adaptive segmentation threshold $\theta = \theta(x, y)$. For example, this threshold can be obtained block-wise, but it can give border effects on the blocks' edges. To avoid this problem, the varying threshold can be considered according to some model. Such models can be stated in different ways that can be found in [139], [140].

Another way to solve the segmentation problem is to use some standard clustering techniques, e.g. MacQueen's k-means algorithm [61], discussed in section 2.2.6.1, with the pixel co-ordinates and intensities as clustering features, or, the same way, by any other general purpose clustering algorithms like those described in this thesis.

Alternatively, one can consider either homogenous regions or contrast edges within the image. For region-based segmentation methods criteria of regions homogeneity and grouping need to be composed, for example hue, gradient, intensity homogeneity, their combination or other criteria which sound appropriate for the practical problems. The remarkable methods, based on region analysis, are region merging [125] and splitting [126], as well as their combination referred as the split-and-merge method [127]. Region merging starts from small regions, typically containing one pixel, and merge it according to some criteria, whilst region splitting, in opposite, decomposes the large regions into subregions until the completion of the stopping criteria.

Watershed segmentation [128] also relies on the regions' homogeneity, but it uses the original idea to consider the image as a surface. One can imagine that this surface is gradually filled by 'water'. Technically it means the gradual increase of the threshold with continuous inspection on the areas which are not 'flooded', i.e. are above the threshold. All continuous areas above the 'water' are referred as 'basins'. Those basins which persist for a long time are considered as segments. Despite one can mind some sorting techniques to make this algorithm working effectively, generally this algorithm is considered as computationally demanding that cannot be factored out for real-time applications.

On the other hand, one can base segmentation on edge analysis. For this purpose, various edge detection methods can be considered, including Canny [129] and Sobel-Feldman [62] algorithms. After edge detection all the continuous areas within the highly discernible edges can be labelled [130]. Another way, the image segmentation problem can be solved by searching for curves around the (hypothetical) objects within an image, which can be expressed by the global optimisation problem. This idea is known as the geometrical curve evolution procedure, which aims to find curves, enveloping segments, given some constraints [131],

[132], [133]. Snake model, for example, uses gradients to extract the area [134] that may cause problems when the intensity changes gradually, and the gradient curve never pass zero [123].

Another group of segmentation algorithms is Markov Random Field (MRF) [135] based methods. The image can be interpreted as a Markov blanket [136], where every pixel is dependent from its neighbours, and the pixels' intensities are conditionally independent of any pixels but neighbours, given neighbours. This Markov blanket can be described by the set of vertices V, representing pixels, and the subset D of $V \times V$ for edges, which gives the relation of neighbourhood. Each of the vertices v_i can be labelled by exactly one value l_i from the label set L. Then it is possible to assign probabilities to the vertices for each of the possible features (usually intensities or any other features depending of the domain), as well as to define a joint event for the neighbouring pixels' features. Then, the special functional over the image pixels' features can be built incorporating both pixels' features probability (unary potential) and joint neighbouring pixels features probability (pairwise potentials). This functional is referred as an Energy Functional and can be written as

$$E(V, D, L, \Pi) = \sum_{v_i \in V} \Phi(v_i, l_i, \Pi) + \sum_{(v_i, v_j) \in D} \Psi\left((v_i, v_j), \Pi\right). \tag{100}$$

Here $\Phi(v_i, l_i, \Pi)$ is a unary potential, and $\Psi\left((v_i, v_j), \Pi\right)$ is a pairwise potential. There are special methods of the optimisation of such functionals, based on graph cuts [164].

An alternative group of methods, also based on the functional optimisation, is given by Mumford-Shah [137] and Chan-Vese [123] functionals. Prior to the formulation of the functionals some definitions are needed. A closed domain for the image is denoted as $\Omega \subset \mathbb{R}^2$, image is defined as $I(x, y): \Omega \to [0, 1]$, $\psi(s): [0, 1] \to \mathbb{R}^2$ is a parametric curve, $s \in [0, 1]$. Let also $\psi(s)$ be a boundary for some region, and $w(x, y) : \Omega \to [0, 1]$ is a model image which is built during the segmentation. Based on this notation, Mumford-Shah functional can be formulated [137]:

$$\begin{gathered} \Phi_{MSh}(w, \psi) = \mu \operatorname{Len}(\psi) + \\ + \lambda \int_{\Omega} |I(x, y) - w(x, y)|^2 \mathrm{d}x\mathrm{d}y + \int_{\Omega \setminus \psi} |\nabla w(x, y)|^2 \mathrm{d}x\mathrm{d}y, \end{gathered} \tag{101}$$

where $\mu > 0, \lambda > 0$ are some empirical parameters; Len (ψ) is the curve length. Here the first term is placed to avoid too long border between the segments, the second one represents the difference between the original and the model image, and the third term imposes a restriction on the gradients inside the model image's regions.

Then, consider some restrictions on the function $w(x,y)$

$$w(x,y) = \begin{cases} \operatorname{avg}_{\xi}\left(u(x,y)\right), (x,y) \in \xi, \\ \operatorname{avg}_{\Omega\backslash\xi}\left(u(x,y)\right), (x,y) \in \Omega\backslash\xi, \end{cases} \tag{102}$$

where ξ is one of the segments, and add area components, responsible for the regulation of one of the segment's area, and the functional turns into Chan-Vese functional [137] for image segmentation:

$$\begin{gathered} F\,(u,\xi) = \mu \operatorname{Len}\left(\psi(\xi)\right) + \nu \operatorname{Area}\,(\xi) + \\ + \lambda_1 \int\limits_{\xi} |u\,(x,y) - w(x,y)|^2 \mathrm{d}x\mathrm{d}y \; + \; \lambda_2 \int\limits_{\Omega\backslash\xi} |u\,(x,y) - w(x,y)|^2 \mathrm{d}x\mathrm{d}y, \end{gathered} \tag{103}$$

where $\operatorname{Area}\,(\xi)$ denotes the area of the segment ξ.

In both cases, the functionals are the subject of minimisation. In this research this functional has been modified as well as the new optimisation scheme was proposed (see section 4.4). Also some link between the proposed solution and MRF has been studied [22].

2.2.8 Template matching techniques

In this section, the following problem is considered: given image I, locate one or multiple objects matching the template J. This template can be represented by an image of the object, or by its descriptor in some feature space, or by any other way, conforming to the problem formulation.

To deal with this problem it is possible to follow several approaches. First it is possible to directly calculate the correlation between the image parts and the template. However, this approach is severely restricted, as it is not capable of pattern generalisation and should be applied multiple times if several different object appearances are considered. To avoid this, one can consider using feature engineering in order to transform both the image fragment and the pattern prior to matching them. This approach can use general-purpose feature engineering techniques like PCA [145], but also domain-specific feature descriptors for graphical information, such as, for example, SIFT [48] for points, MSER for areas [66]. Another way is to change the way to match patterns. Instead of the correlation, some complex classifiers for pattern matching can be applied, e.g. boosted trees for Haar-type features used in Viola-Jones algorithm [8], mentioned in section 2.2.3. The feature descriptors are discussed in the section 2.2.9.

The following interpretation of the pattern matching, based on the statistical hypothesis check, can be discussed. One can formulate two statements, null- and alternative hypothesis, for I:

- the object is not present in the given fragment of the image(H_0);
- the object is present in the given fragment of the image(H_1).

The discrimination between these contradictive statements can be interpreted as a hypothesis testing. One can identify an image I in terms of interfering signals and represent the hypothesis as:

$$H_0: I(x,y) = \zeta(x,y), \tag{104}$$

$$H_1: I(x,y) = \zeta(x,y) + J(x,y). \tag{105}$$

Here $\zeta(x,y)$ is clutter, i.e. something not representing the object we search for. The verification of the hypotheses can be implemented using covariance:

$$K_i(f_i) = \int\limits_{(x_j,y_j)\in\Omega_J} \left\{I\left(f_i(x_j,y_j)\right) - \mu_I\right\}\{J(x_j,y_j) - \mu_J\}dx_j dy_j\,, \tag{106}$$

with mean values defined as

$$\mu_I = E\left[I\left(f(x_j), f(y_j)\right)\right], \tag{107}$$

$$\mu_J = E[J(x_j,y_j)]. \tag{108}$$

Here Ω_J is a domain for the pattern J, $f_i(x_j,y_j)\colon \Omega_J \to \Omega_I$, Ω_I is a domain of the image I (both I and J can be either in the same or in different spaces).

A more complex model provides alternative decomposition:

$$H_0: I(x,y) = \zeta(x,y), \tag{109}$$

$$H_1: I(x,y) = \alpha J(x,y) + \beta + \zeta(x,y). \tag{110}$$

Here α and β are some parameters for the intensity shift and scaling.

In order to get the result of pattern matching from the hypothesis, the normalisation of both pattern J and the image I before the covariance estimation. One of the simplest approaches to deal with it is to utilise normalised correlation coefficient in order to remove shift and scatter:

$$K(f_i = f_i(x_j,y_j)) =$$

$$= \int\limits_{\substack{(x_j,y_j)\\ \in\Omega_J}} \frac{\{I(f_i) - \mu_I\}\{J(x_j,y_j) - \mu_J\}dx_j dy_j}{\sqrt{\left[\int_{(x_j,y_j)\in\Omega_J}\{I(f_i) - \mu_I\}^2 dx_j dy_j\right]\left[\int_{(x_j,y_j)\in\Omega_J}\{J(x_j,y_j) - \mu_J\}^2 dx_j dy_j\right]}} \tag{111}$$

The normalised correlation coefficient can be interpreted as a cosine similarity for the vectors x, y:

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{|\boldsymbol{x}||\boldsymbol{y}|}, \tag{112}$$

where $x = I\big(f_i(x_j, y_j)\big) - \mu_I,\ y = J\big(x_j, y_j\big) - \mu_J$.

The most common way to select a mapping of the pattern to the image is the sliding window:

$$f_i\big(x_j, y_j | x, y\big) = \big(x + x_j, y + y_j\big) \colon \big(x_j, y_j\big) \in \Omega_J, \big(x + x_j, y + y_j\big) \in \Omega_I. \tag{113}$$

The testing procedure can be done for all (x, y) giving $\big(x + x_j, y + y_j\big) \in \Omega_I\ \forall \big(x_j, y_j\big) \in \Omega_J$.

However, this method itself cannot give the universal robust approach to object localisation. Neither does it take into account distortion nor scale. To increase the robustness to noise it is necessary to use more complicated methods like proposed in [138], where feature extraction techniques invariant to the transform and noise are used. Some of the feature extraction methods, which can be used jointly with pattern matching, are given in the following section. Additionally, one can consider using more sophisticated criteria for each of the sliding windows, e.g. using Viola-Jones algorithm [8], mentioned in section 2.2.3.

2.2.9 Feature extraction survey

The dimensionality of the data space can be prohibitively large comparing to the memory and time resources available on the computer. For example, single image contains thousands or even millions of deeply correlated pixels. Therefore, there is a strong need for extraction of the compact data description, in particular for graphical data, capable of significant reduction of the data vector dimensionality and of removing the component-wise correlations. Due to all these reasons, the feature extraction techniques are of paramount importance for the pattern recognition problems. Though using different approaches, all these method are united by the common aim, namely to extract the informative features (w.r.t. some criteria) and represent them in compact vectors with lower intercomponent correlation, reducing the vectors' dimensionalities. Some of the approaches are general and are defined to minimise loss expressed in some pre-defined functional, as in PCA [190] and some similar techniques. Other are domain specific, related to the image analysis, as corner detection and image feature descriptors. The techniques of tree boosting, mentioned in section 2.2.3, can also be used as general purpose feature extractor techniques. Ad hoc methods are not highlighted, because they are mostly proposed for particular problems and, therefore, are out of the scope of this thesis.

2.2.9.1 *PCA*

In this section the rigorous method for feature extraction and dimensionality reduction, Principal Component Analysis (PCA), is briefly discussed. The method was first proposed by Karl Pearson in 1901 [190]. It has led to a broad class of methods: Independent Component Analysis (ICA) [141], 2D PCA [142], Kernel PCA [143], probabilistic PCA [145], and many others.

Define a space $\mathbb{R}^K$ with an orthogonal K-dimensional basis $\{\boldsymbol{v}_i\}, i = 1 \dots K$, $\boldsymbol{v}_i^T \boldsymbol{v}_j = \delta_{ij}$, where δ_{ij} is a Kronecker operator

$$\delta_{ij} = \begin{cases} 1, i = j, \\ 0, \text{else}. \end{cases} \tag{114}$$

Then K-dimensional points from the set $X = \{\boldsymbol{x}_1 \dots \boldsymbol{x}_N\}$ can be represented in terms of the basis $\{\boldsymbol{v}_i\}$ [144]:

$$\boldsymbol{x}_n = \sum_{i=1}^{K} a_{ni} \boldsymbol{v}_i, \tag{115}$$

where

$$\boldsymbol{a}_{ni} = x_n^T \boldsymbol{v}_i \tag{116}$$

can be considered as a co-ordinate system transformation.

There are infinitely many ways to transform the co-ordinate system. The PCA aims to find such transformation, which delivers the most accurate projection (in the sense of some function under optimisation, which we discuss further) into the dimensionality $M \leq K$. For this purpose, the original vector $\boldsymbol{x}_n$ is approximated by the following one:

$$\widehat{\boldsymbol{x}}_n = \sum_{i=1}^{M} a_{ni} \boldsymbol{v}_i + \sum_{i=M+1}^{K} b_i \boldsymbol{u}_i, \tag{117}$$

where $\{a_{ni}\}$ are dependent of the vector we approximate and are the same as given by the equation (116), and b_i are some fixed quantities. The following optimisation problem can be formulated to select such transformation:

$$J(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \widehat{\boldsymbol{x}}_1, \dots, \widehat{\boldsymbol{x}}_n) = \frac{1}{N} \sum_{n=1}^{N} \|\boldsymbol{x}_n - \widehat{\boldsymbol{x}}_n\|^2 \to \min_{a,b}, \tag{118}$$

$$\sum_{i=1}^{K} \boldsymbol{v}_i^T \boldsymbol{v}_i = 1, \boldsymbol{v}_i^T \boldsymbol{v}_j = \delta_{ij}, i, j \in [1 \dots K], \tag{119}$$

where $||\boldsymbol{x}_{\mathrm{n}} - \widehat{\boldsymbol{x}}_{\mathrm{n}}||$ is a norm operator. Taking the derivative, one can obtain the following solution:

$$a_{ni} = \boldsymbol{x}_n^T \boldsymbol{v}_i, \tag{120}$$

$$b_i = \left(\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i \right)^T \boldsymbol{v}_i. \tag{121}$$

After the substitution of the solution to the optimisation problem one can obtain

$$J(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \widehat{\boldsymbol{x}}_1, \dots, \widehat{\boldsymbol{x}}_n) = \\ = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{K} \left\| \boldsymbol{x}_n^T \boldsymbol{v}_i - \left(\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i \right) \boldsymbol{v}_i \right\|^2 = \sum_{i=M+1}^{K} \boldsymbol{v}_i^T S \boldsymbol{v}_i, \tag{122}$$

where S is the covariance matrix.

The final formulation of the optimisation problem (118), (119) can be given as follows:

$$J(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n, \widehat{\boldsymbol{x}}_1, \dots, \widehat{\boldsymbol{x}}_n) \to \min_v , \tag{123}$$

$$\sum_{i=1}^{K} \boldsymbol{v}_i^T \boldsymbol{v}_i = 1. \tag{124}$$

It can be proven that the general minimisation problem solution consists of the eigenvectors, which are taken from the eigenvector equation

$$S\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i, i = 1 \dots K. \tag{125}$$

The minimal value is reached, when M principal vectors with the largest eigenvalues are selected, hence leaving $\boldsymbol{v}_i, i = (M+1) \dots K$ the eigenvectors with the smallest eigenvalues. Then the co-ordinate system is transformed to $\{\boldsymbol{v}_i\}, i = 1 \dots K$ with omission of the last $K - M$ components of the vector.

2.2.9.2 *Corner detectors and descriptors*

Due to the extensive development of the image and video analysis algorithms the following problems are important:

- concise description of the image content, reflecting its specific features, resulting in the vectors with low-correlated components;
- image comparison, widely used for video tracking, object detection and stereo-matching.

For both these problems there is a need to find out those points or regions of the image which contain much information about the image content. For example, they can be image corner points, which can be described by specific score, based on the change of the image intensity. One of the first methods, implementing this idea, was Moravec's corner detector [146]. This detector uses the following simple score for each pixel (x, y) of the image I to be a corner:

$$E(x,y) = \frac{1}{\sum_{u,v} w(u,v,x,y)} \sum_{u,v} w(u,v,x,y)[I(x,y) - I(u,v)]^2, \qquad (126)$$

where $(x, y), (u,\ v)$ are the image pixels, and $w(u, v, x, y)$ is the function, turning to 1, if the pixels $(x, y), (u,\ v)$ are within the neighbourhood area, and 0 otherwise. As a result of calculations, the corner score map is obtained. Applying local optimisation with non-maximum suppression, i.e. elimination of the local maxima in the vicinity of the larger local maximum, one can obtain the corner points list. However, this model is oversimplified for many real problems and its drawback is that the method's performance severely depends of gradient direction in the point's neighbourhood.

To address practical problems, Harris-Stephens corner detector [191] was proposed as an enhanced version of the Moravec corner detector. In this detector, Boolean characteristic function $w(u, v, x, y)$ is replaced by the Gaussian operator with the standard deviation parameter σ:

$$w(u,v,x,y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(u-x)^2 + (v-y)^2}{2\sigma^2}\right). \qquad (127)$$

After that, a Taylor approximation is calculated in the neighbourhood of the image pixels:

$$I(x + \Delta x, y + \Delta y) \approx I(x,y) + \frac{\partial I(x,y)}{\partial x}\Delta x + \frac{\partial I(x,y)}{\partial y}\Delta y. \qquad (128)$$

After substitution of this expression to the equation (126) one can obtain Harris matrix with the same dimensions as the source image as follows:

$$E(x,y) = \sum_{u,v} w(u,v,x,y) \left[\frac{\partial I(x,y)}{\partial x}(u-x) + \frac{\partial I(x,y)}{\partial y}(v-y)\right]^2, \qquad (129)$$

This score can be also written as $E(x, y) = (x\ y)T(x\ y)^T$ where T is the weighted partial derivative matrix. The original score for the point being corner is based on the eigenvalue decomposition for the matrix T for all the candidate pixels. This circumstance hinders the real time algorithm applications. Therefore, some easier criterion is needed. This criterion, giving approximately the same result due to heuristic reasoning, was formulated [191] as

$$R(u,v) = \det\begin{bmatrix} \frac{\partial^2 I(u,v)}{\partial^2 u} & \frac{\partial^2 I(u,v)}{\partial u \partial v} \\ \frac{\partial^2 I(u,v)}{\partial u \partial v} & \frac{\partial^2 I(u,v)}{\partial^2 v} \end{bmatrix} - k\,\mathrm{tr}^2\begin{bmatrix} \frac{\partial^2 I(u,v)}{\partial^2 u} & \frac{\partial^2 I(u,v)}{\partial u \partial v} \\ \frac{\partial^2 I(u,v)}{\partial u \partial v} & \frac{\partial^2 I(u,v)}{\partial^2 v} \end{bmatrix}, \tag{130}$$

where k is a parameter.

Based on the Harris corner detector ideas, more complicated detectors were developed. They aim not only to find the points according to the score, but also to describe them in the way allowing their matching on different images for the purposes of recognition, tracking and detection.

For example, Scale Invariant Feature Transform (SIFT) algorithm contains the following stages [48]:

1) A Gaussian Pyramid is built upon the grey-scale image by alternating stages of smoothing and subsampling, then their differences are calculated resulting in Difference of Gaussians (DoG) model;
2) In the way, highly resembling Harris operator, the score is composed, which suppresses the edges and low gradient areas and boosts the corner-like points;
3) Then, the output of the previous stage is thresholded, and the points with the strongest outputs are selected.
4) For each of the selected points the local histogram of oriented gradients is built, and the prevailing direction is determined. The size of consistent area is also determined for each interest point [147].
5) The local histogram of oriented gradients is enhanced, and normalized to constant sum. What appears is referred as an interest point descriptor. After then, SIFT descriptors can be matched for different images by Nearest Neighbours matching procedure, or treated by any other machine learning algorithm.

There are several descriptors, resembling SIFT by structure but having some differences in the used heuristics at each of the stage, such as Speeded Up Robust Features (SURF) [148] and Gradient Location and Orientation Histogram (GLOH) [149]. Also, the Histogram of Oriented Gradients (HOG) region descriptor [150] is inspired by the local histogram of oriented gradients, featured is SIFT [192]. Some descriptors use an alternative structure and ideas, like gist descriptor [182] based on Gabor wavelet features [193]. Such variety of methods can be explained by highly complex heuristic feature extraction method structure, high importance of the practical problems addressed by the methods. Another idea is that the regions can be described instead of points like in MSER descriptor [66]. All these methods provide information which can be used both for description of the local features of the images and their between-image matching. Also these methods can be useful for the whole image feature description that can be particularly useful for subsequent clustering and/or classification of the images.

2.3 Conclusion

Reflecting the twofold nature of the topic of the thesis, this chapter contains separate surveys for the object tracking, and the detection and recognition methods, both emphasising the historical roots of the variety of the state-of-the-art methods.

The object tracking problems and their solutions, both for military and civil applications, have been attracting researchers from time immemorial; however, the present state of this area has been enormously affected during the last sixty years by pervasive application of the computing machines. When, during the last decades, the computational powers became sufficient for video processing, it became one of the most valuable applications of object tracking. Section 2.1 describes the various methods of object tracking, starting from the nearest neighbour approach, and then splitting the video tracking methods by two large groups: Bayesian and non-Bayesian tracking. The Bayesian filter methods are usually general purpose, i.e. they allow parameterisation for the particular problem and might not be restricted to video sensors, and are all based on the same graphical model [35]. These methods include Kalman filter [37], particle filters [40], as well as target association models. The non-Bayesian methods include trackers designed for the video applications and can rely on video-specific methods such as optical flows. The given examples of such methods include TLD [26] and ARTOT [47]. One of the approaches, relevant to this thesis, is 'rigid motion segmentation' [42], which inspired the author on the model, proposed in chapter 3, and aims to build a time-consistent segmentation of videos on similarly moving areas. The review of the object tracking methods is finished by an optical flow concept review [60], which is a necessary component of many visual tracking methods.

The object detection and recognition problems, both in general and in application to the pictorial data, are described in section 2.2. This area shown even more diverse groups of methods than tracking, and is reviewed from different points of view. The review embraces such diverse areas, related by the stated problems, as neural networks, SVMs, fuzzy logic, and focuses on such standard data mining problems as clustering and classification. The technical details, which are relevant to the following narrative, are given for k-means clustering [61], mixtures of Gaussians and the EM algorithm [120]. For image segmentation techniques, which arise from the clustering problem, the technical details are given for the Markov Random fields [135] and graph cut[164] methods, and the allied Mumford-Shah [137] and Chan-Vese [123] functionals. In order to cover the area of the detection, widely renowned template matching techniques are briefly reviewed. The necessary part of the contemporary methods for object detection and recognition is feature extraction. In order to cover this area in survey, the widely known PCA method [190] is reviewed, as well as the image-specific corner descriptors, which are used in the description of the proposed video tracking method.

3 Proposed object tracking techniques

Most of the object tracking methods, described in the survey in section 2.1, can be assigned to one of the following two groups. Algorithms from the first group are based on theoretically well-grounded Bayesian filtering framework. This framework is independent of the tracking problem domain and can be implemented for particular object models and application domains. In particular, such a generalised tracking model can be applied to video data. Second group of algorithms is based on domain-specific models, which are designed for specific tracking use case. The excellent example of such algorithm is the Tracking-Learning-Detection (TLD) algorithm [34], designed for the particular case of single object tracking with full or partial overlapping given that the object appearance and position is similar in the neighbouring frames.

In this research, a novel multiple object detection and tracking technique, based on the Bayesian filtering framework, is proposed. Contrary to many Bayesian filters for multiple object tracking [151], in the method, proposed in this research, there is no assumptions whether the measurements are generated by clutter or by target. Instead of this, the filter is used for time-consistent clustering of data, and the objects of interest are selected from these clusters. Because of this construction, the method also resembles well-known works on rigid motion segmentation [152].

3.1 Practical motivation of the method

Consider the video as a series of images $\{I_1, I_2,\ I_k, \dots\}$, received from the camera, where k can be considered as a time index. It is supposed that the camera can move, and there are objects in the camera's area of view, moving independently of the camera (ships, cars, planes or any other moving objects).

The problem is to track the independently moving image segments. Here to track means to build a correspondence between the parts of the image within the frames I_k and I_{k+1} for any k and $k+1$. From the practical point of view, it is also needed to emphasise that the problem solution should not rely on training but provide a general motion segmentation model. This problem statement is in line with the tracking problems described in the section 2.1, containing the object tracking methods survey. This problem can be described as 'time-consistent clustering'.

Although the method, proposed in this thesis, is applied to the video analytics domain, its formulation is general as the Bayesian filtering framework does not impose restriction on the nature of data used within the inference framework; additionally, as clustering is one of the general problems of data mining, the applications may expand beyond the video analytics, and even tracking area.

The approach, described in this section, includes the novel domain independent Bayesian filter for object tracking. The idea of the method, as well as its difference from the state-of-the-art methods, is in combination of two different concepts: Bayesian filtering, derived from signal processing perspective, and time consistent clustering, which is a variation of the classical data mining problem of clustering. Apart from the many well-known models, the method does not divide the clutter and objects into a tracking stage, but tracks them all simultaneously. The selection of the object clusters is deferred to the subsequent domain-specific object detection stage.

The proposed solution has been applied to video analysis the following way. The algorithm is continuously clustering of the set of points of the video, evolving in time; the clusters are continuously tracked. At this stage it is assumed that the majority of the points moves according to the background velocity model, which incorporates linear and angular velocity. Based on this model, it is possible to factor out those clusters, which conform to the background according to the model, and select only those which are moving. It is particularly useful for the motion detection for a moving camera rather than static. The proposed algorithm is also capable of performing in real time or near real time on standard PCs as it is detailed further during the experiments descriptions.

The method can be complemented by the geographical co-ordinates estimation methods (see section 3.6). Using the assumption that the objects are laying on the planar surface, it is possible to estimate the distances and calculate the geographical co-ordinates given the geographical co-ordinates of filming sensor and data fusion sensors.

3.2 Bayesian filter based algorithm for Gaussian mixture propagation

In this chapter, the domain-independent Bayesian filter model for time consistent clustering is proposed [15]. This approach implies that there is no differentiation between the clutter and the objects until the detection stage. The description starts with the most general formulation of the Bayesian filter [153], which is then parameterised [15]

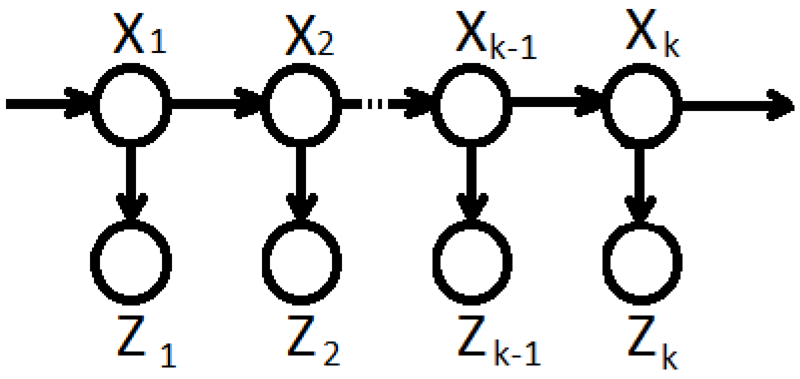


Figure 6. The graphical model for a Bayesian filter

The Bayesian model in Figure 6 is given by the interconnected hidden states X_k and visible states Z_k, where $k \geq 1$ is often references as a scan number.

The posterior probability of the state depending on the measurements is found using the Bayesian rule simplified thanks to the assumptions on the current state's conditional independence of the previous states except the last one that is reflected in Figure 6:

$$p(X_k|Z_1 \dots Z_k) = \frac{p(Z_k|Z_1 \dots Z_{k-1}, X_k)p(X_k|Z_1 \dots Z_{k-1})}{p(Z_k, X_k|Z_1 \dots Z_{k-1})} = \\ = \frac{p(Z_k|X_k)p(X_k|Z_1 \dots Z_{k-1})}{p(Z_k, X_1 \dots X_k|Z_1 \dots Z_{k-1})} \propto p(Z_k|X_k)p(X_k|Z_1 \dots Z_{k-1}). \quad (131)$$

The proportionality coefficient can be derived from the probability normalisation condition.

Usually, the Bayesian filters are logically divided into two steps: i) prediction, and ii) update. The prediction step is expressed as

$$p(X_k|Z_1 \dots Z_{k-1}) = \int p(X_k|X_{k-1})p(X_{k-1}|Z_1 \dots Z_{k-1})dX_{k-1}. \quad (132)$$

In this statement the integration sign can be replaced by the sum for the discrete distribution. The update step is calculation of the posterior distribution according to formula (131). Altogether, it can be seen as a recursion, as the posterior distribution $p(X_{k-1}|Z_1 \dots Z_{k-1})$ derived from the previous stage update step (except the first step, where the initial probability needs to be defined separately as a part of the model) is used for the prediction step at the next stage.

To define a Bayesian filter, one needs to define the hidden and visible variables and to assign initial $p(X_1|Z_1)$, prior $p(Z_k|X_k)$ and transitional $p(X_k|X_{k-1})$ probabilities. Each of the visible variables in the proposed model is the measurement vectors set (i.e. co-ordinates and velocities of the objects). The hidden variables are the parameters of the Gaussian components within the Gaussian mixture (namely centre and covariance for each of the components, as well as their weights). More formally, the object features set $Z_k = \{\boldsymbol{z}_1^k, \boldsymbol{z}_2^k \dots \boldsymbol{z}_{n_k}^k\}$ is considered as the visible variables, and the parameters of the Gaussians within the Gaussian mixture: means $\boldsymbol{\mu}_k = \{\boldsymbol{\mu}_1^k, \boldsymbol{\mu}_2^k \dots \boldsymbol{\mu}_K^k\}$, covariance matrices $\Sigma_k = \{\Sigma_1^k, \Sigma_2^k \dots \Sigma_K^k\}$, and weights $\pi_k = \{\pi_1^k, \pi_2^k \dots \pi_K^k\}$ are hidden variables. The problem, as with any Bayesian filter, is to define the most probable hidden configuration for the given measurements. The assignment of the initial, prior and transitional probabilities for maintaining the parameters of the Gaussian mixtures in a time-consistent way in a Bayesian filter framework is described in the following subsections. The initial probability assignment is discussed in section 3.2.1, the prior probabilities are described in section 3.2.2, and the transitional probabilities are discussed in section 3.2.3. The Bayesian filter recursion solution for the update step is described in section 3.2.4.

3.2.1 System initialisation

In this section, the Bayesian recursion initialisation is described. The plain Gaussian Mixture EM algorithm is performed just as it is described in the section 2.2.6.1 to get the parameters π_i^1, $\boldsymbol{\mu}_i^1$, Σ_i^1, and as a result the initial probability is obtained in the form of mixture of Gaussians:

$$p(z_i^1) = \sum_{i=1}^{K} \pi_i^1 \mathcal{N}(z_i^1 | \boldsymbol{\mu}_i^1, \Sigma_i^1). \tag{133}$$

We assume the parameters to have the following distributions (and further in the text this distribution form will be preserved for all the subsequent states):

$$\begin{gathered} \boldsymbol{\mu}_i^1 \sim \mathcal{N}(\boldsymbol{\mu}_i^1 | m_i^1, \xi_i^1), \\ \Lambda_i^1 = [\Sigma_i^1]^{-1} \sim \mathcal{W}(\Lambda_i^1 | W_i^1, \nu_i^1), \\ \pi^1 \sim \mathrm{Dir}\,(\pi^1 | \lambda^1), \\ i = 1 \dots N, \end{gathered} \tag{134}$$

where $\mathcal{W}(\cdot)$ stands for the Wishart distribution, and $\mathrm{Dir}(\cdot)$ stands for the Dirichlet distribution. At the first step, ξ_i^1 are initialised to some pre-defined value (e.g. proportional to the identity matrix), and λ^1 (and, as it can be seen further in the text, $\lambda^k, k \geq 1$ for the subsequent stages) are set to high values in order to make Dirichlet distribution variance for each of the weights lower.

3.2.2 Prediction

In this section, the description of the proposed prediction model is described [15]. The prediction step exploits the information of the feature points' movement w.r.t. the previous frame. In this part of the algorithm, the aim is to estimate the prediction $p(X_k|Z_1 \dots Z_{k-1})$. The equation (132) is parameterised in following way:

$$p(X_k|Z_1 \dots Z_{k-1}) = \\ = \int p(\boldsymbol{\mu}^k, \Lambda^k, \pi^k | \mu^{k-1}, \Lambda^{k-1}, \pi^{k-1}) p(\boldsymbol{\mu}^{k-1}, \Lambda^{k-1}, \pi^{k-1} | Z_1 \dots Z_{k-1}) d\boldsymbol{\mu}^{k-1} d\Lambda^{k-1} d\pi^{k-1}, \quad (135)$$

where $\Lambda_i^k = \left[\Sigma_i^k\right]^{-1}$.

The distribution $p(X_{k-1}|Z_1 \dots Z_{k-1})$ is built on the assumption that it is factorised by each of the Gaussian distribution parameters (that is true for the first stage and it also assumed for the results from the update stage in section 3.2.3) and hence can be represented as follows:

$$p(X_k|Z_1 \dots Z_{k-1}) = \\ = \left[\prod_{i=1}^{K} \left(p\left(\boldsymbol{\mu}_i^k | Z_1 \dots Z_{k-1}\right) p\left(\Lambda_i^k | Z_1 \dots Z_{k-1}\right)\right)\right] p(\pi^k | Z_1 \dots Z_{k-1}). \quad (136)$$

Then [15]

$$p\left(\boldsymbol{\mu}_i^k | Z_1 \dots Z_{k-1}\right) = \int p\left(\boldsymbol{\mu}_i^k | \mu_i^{k-1}\right) p\left(\boldsymbol{\mu}_i^{k-1} | Z_1 \dots Z_{k-1}\right) d\mu_i^{k-1}, \quad (137)$$

$$p\left(\Lambda_i^k | Z_1 \dots Z_{k-1}\right) = \int p\left(\Lambda_i^k | \Lambda_i^{k-1}\right) p\left(\Lambda_i^{k-1} | Z_1 \dots Z_{k-1}\right) d\Lambda_i^{k-1}, \quad (138)$$

$$p(\pi^k | Z_1 \dots Z_{k-1}) = \int p(\pi^k | \pi^{k-1}) p(\pi^{k-1} | Z_1 \dots Z_{k-1}) d\pi^{k-1}, \quad (139)$$

For the mean, it is possible to assume

$$p\left(\boldsymbol{\mu}_i^k | \boldsymbol{\mu}_i^{k-1}\right) = \mathcal{N}\left(\boldsymbol{\mu}_i^k | \widetilde{\boldsymbol{m}}_i^k, \tilde{\Gamma}_i^k\right). \quad (140)$$

can be interpreted as a new object position estimation, based on the affine transformation of the previous value $\boldsymbol{\mu}_i^{k-1}$ using the rotation matrix R^k, and translation vector T^k, estimated between the previous and the new measurement sets, and $\tilde{\Gamma}_i^k$ is the covariance matrix for the new model position estimation. Here, the predictive distribution parameters are determined by the linear model, exploiting the results from the Kalman filter:

$$p\left(\boldsymbol{\mu}_i^k | Z_1 \dots Z_{k-1}\right) \sim \mathcal{N}\left(\boldsymbol{\mu}_i^k | \widetilde{\boldsymbol{m}}_i^k, \tilde{\xi}_i^k\right), \quad (141)$$

$$\widetilde{\boldsymbol{m}}_i^k = R^k \boldsymbol{\mu}_i^{k-1} + T^k, \quad (142)$$

$$\tilde{\xi}_i^k = R^k \xi_i^{k-1} [R^k]^T + \tilde{\Gamma}_i^k. \quad (143)$$

The parameter R^k and T^k are determined from the following least squares method equations:

$$\operatorname{tr}\left(R^k Z_{k-1} + (T^k \dots T^k)_{n_k} - Z_k\right)^T \left(R^k Z_{k-1} + (T^k \dots T^k)_{n_k} - Z_k\right) \to \min_{R^k, T^k}, \quad (144)$$

where Z_k are represented as measurement matrices, where the columns correspond to the elements of the measurements set, and the rows to the vectors, describing these measurements.

It can be seen from here that R^k is a square orthogonal matrix (i.e. $[R^k]^T = [R^k]^{-1}$) , and T^k is a translation matrix with the same size as a feature set dimensionality

$$\tilde{\Gamma}_i^k = \operatorname{cov}\left[R^k Z_{k-1} + (T^k \dots T^k)_{n_k} - Z_k\right]. \tag{145}$$

For other parts of the prediction step, as it involved the Wishart and Dirichlet distributions, in order to obtain a closed form, the separate assignments of the transition probabilities were replaced by the following predictions:

$$p(\Lambda_i^k | Z_1 \dots Z_{k-1}) \sim \mathcal{W}(\Lambda_i^k | W_i^k, \nu_i^k), \tag{146}$$

$$p(\pi^k | Z_1 \dots Z_{k-1}) \sim \operatorname{Dir}(\pi^k | \lambda^k), \tag{147}$$

Here for the inverse covariance matrix $W_i^k = W_i^{k-1}, \nu_i^k = \nu_i^{k-1}$. For $\operatorname{Dir}(\pi^k|\lambda^k)$ the parameters $\lambda^k = (\lambda_1^k \dots \lambda_K^k)$ are chosen high enough to effectively equalise the priors for the weight parameters π^k to the values π^{k-1} as it were in the previous stage.

3.2.3 Update

In this section, the update equations are defined. Once $p(X_k|Z_1 \dots Z_{k-1})$ is derived, one can obtain expressions for $p(Z_k|X_k)$. Using the formula (131), it is possible to find $p(X_k|Z_1 \dots Z_k) = p(Z_k|X_k) p(X_k|Z_1 \dots Z_{k-1})$ using the maximum likelihood criterion.

First, $p(Z_k|X_k)$ needs to be defined. It is given by the following factorisation over the elements of the measurement set, where for each data sample the distribution is defined as a Gaussian mixture:

$$p(Z_k|X_k) = \prod_{i=1}^{n_k} p(\boldsymbol{z}_i^k | X_k), \tag{148}$$

$$p(\boldsymbol{z}_i^k | X_k) = \sum_{j=1}^{K} \pi_j^k \mathcal{N}\left(\boldsymbol{z}_i^k \middle| \boldsymbol{\mu}_j^k, [\Lambda_j^k]^{-1}\right). \tag{149}$$

Then one can write the complete-data likelihood as

$$p(X_k|Z_1 \dots Z_k) = \\ = \prod_{i=1}^{n_k} \left[\sum_{j=1}^{K} \pi_j^k \mathcal{N}\left(z_i^k \middle| \boldsymbol{\mu}_j^k, [\Lambda_j^k]^{-1}\right)\right] \prod_{i=1}^{K} \left[\mathcal{N}(\boldsymbol{\mu}_i^k | \widetilde{\boldsymbol{m}}_i^k, \xi_i^k)\, \mathcal{W}(\Lambda_i^k | W_i^k, \nu_i^k)\right]. \tag{150}$$

Here, the Dirichlet distribution is approximated over π^k by a δ-function, using the assumption of the large values of Dirichlet distribution parameters given in section 3.2.2.

Then one can write the log-likelihood and state the following optimisation problem:

$$\log p(X_k|Z_1 \dots Z_k) = \sum_{i=1}^{n_k} \log \left[\sum_{j=1}^{K} \pi_j^k \mathcal{N}\left(z_i^k \middle| \boldsymbol{\mu}_j^k, [\Lambda_j^k]^{-1}\right) \right] +$$
$$+ \sum_{i=1}^{K} \left[\log \mathcal{N}\left(\boldsymbol{\mu}_i^k \middle| \widetilde{\boldsymbol{m}}_i^k, \tilde{\xi}_i^k\right) + \log \mathcal{W}\left(\Lambda_i^k \middle| W_i^k, \nu_i^k\right)\right] \to \max_{\mu^k, \Lambda^k, \pi^k}, \tag{151}$$

$$\sum_{j=1}^{K} \pi_j^k = 1. \tag{152}$$

The optimisation of this function is carried out by the EM algorithm, briefly described in section 2.2.6.1. However, as it is seen from the section 3.2.4, where the derivation of the EM algorithm for this particular case is given, the posterior distribution form deviates from that given by the Gaussian mixture model (GMM). Therefore, the new posterior distribution parameters should be calculated by some approximation. In the proposed Bayesian algorithm implementation Laplace approximation is used for such approximation.

This approximation preserves the mean of the distribution $\boldsymbol{m}_i^k = \boldsymbol{\mu}_i^k, i = 1 \dots K,$ as it is assigned to the mode, and the covariance is expressed as a negative inverse Gaussian of the maximum posterior likelihood:

$$\Xi_i^k = -\left[\nabla_{m_i^k} \nabla_{m_i^k} \log(\log p(\boldsymbol{\mu}^k, \Lambda^k, \pi^k | Z_1 \dots Z_k))\right]^{-1}. \tag{153}$$

Here, the sign $[\cdot]^*$ stands for the optimal solution for the optimisation problem (151)-(152). The value of the weights π_i^k are transferred to the posterior without any modifications.

3.2.4 EM algorithm for the proposed model

As it was stated before, the maximum likelihood optimisation cannot be performed analytically, hence the approximation scheme is to be provided. The proposed maximum likelihood scheme is based on the Expectation-Maximisation (EM) algorithm described in section 2.2.6.1. For this purpose, the likelihood is iteratively approximated by easy to optimise mean complete data likelihood, as it was described in section 2.2.6.1:

$$\log p(X_k, Z_k | Z_1 \dots Z_{k-1}) =$$
$$= \sum_{i=1}^{n_k} \left[\sum_{j=1}^{K} \hat{p}\left(x_j^k \middle| \boldsymbol{z}_i^k\right) \left(\log \pi_j^k + \log \mathcal{N}\left(\boldsymbol{z}_i^k \middle| \boldsymbol{\mu}_j^k, [\Lambda_j^k]^{-1}\right)\right)\right] +$$
$$+ \sum_{i=1}^{K} \left[\log \mathcal{N}\left(\boldsymbol{\mu}_i^k \middle| \widetilde{\boldsymbol{m}}_i^k, \tilde{\xi}_i^k\right) + \log \mathcal{W}\left(\Lambda_i^k \middle| W_i^k, \nu_i^k\right)\right] \to \max_{\mu^k, \Lambda^k, \pi^k}, \tag{154}$$

w.r.t. the constraint imposed on π^k:

$$\sum_{j=1}^{K} \pi_j^k = 1. \tag{155}$$

Here $\hat{p}(x_j|\boldsymbol{z}_i)$ is the posterior estimation derived from the E-step of the EM algorithm. After differentiation of the Lagrangian

$$\begin{aligned}\mathfrak{L}(X_k, Z_k|Z_1 \dots Z_{k-1}) = \sum_{i=1}^{n_k} \left[\sum_{j=1}^{K} \hat{p}(x_j^k|\boldsymbol{z}_i^k) \left(\log \pi^k + \log \mathcal{N}\left(\boldsymbol{z}_i^k \middle| \boldsymbol{\mu}_j^k, [\Lambda_j^k]^{-1} \right) \right) \right] + \\ + \sum_{i=1}^{K} \left[\log \mathcal{N}(\boldsymbol{\mu}_i^k|\widetilde{\boldsymbol{m}}_i^k, \xi_i^k) + \log \mathcal{W}(\Lambda_i^k|W_i^k, \nu_i^k) \right] - \lambda \left(\sum_{j=1}^{K} \pi^k - 1 \right),\end{aligned} \tag{156}$$

obtained from the likelihood function and the constraint, local extremum values $\widehat{\boldsymbol{\mu}}^k, \widehat{\Lambda}^k, \hat{\pi}^k$ (M-step) are calculated, given the values of the posterior distribution estimation $\hat{p}(x_j|z_i)$, obtained on the E-step.

The mean, covariance and weights are initialised for the first stage of the EM algorithm by the prediction step mean values:

$$\boldsymbol{\mu}_j^k = \widetilde{\boldsymbol{m}}_j^k, \tag{157}$$

$$\Lambda_j^k = \nu_j^k W_j^k, \tag{158}$$

$$\pi_j^k = \frac{\lambda_j^k}{\sum_{i=1}^{K} \lambda_i^k}. \tag{159}$$

The E-step expressions are obtained using the Bayes' theorem:

$$\hat{p}(x_j^k|z_i^k) = \frac{\pi_j^k \mathcal{N}(\boldsymbol{z}_i^k|\boldsymbol{\mu}_j^k, \Sigma_j^k)}{\sum_{m=1}^{K} \pi_m^k \mathcal{N}(z_i^k|\boldsymbol{\mu}_m^k, \Sigma_m^k)}. \tag{160}$$

$$x_j^k = \{\boldsymbol{\mu}_j^k, \Lambda_j^k, \pi_j^k\} \tag{161}$$

To obtain the M-step expressions, one can write the derivatives for the Lagrangian with respect to all the parameters:

$$\begin{aligned}\frac{\partial\, \mathfrak{L}(X_k, Z_k|Z_1 \dots Z_{k-1})}{\partial \boldsymbol{\mu}_j^k} = \sum_{i=1}^{n_k} \frac{\partial}{\partial \boldsymbol{\mu}_j^k} \left[\hat{p}(x_j^k|\boldsymbol{z}_i^k) \log \mathcal{N}\left(z_i^k \middle| \boldsymbol{\mu}_j^k, [\Lambda_j^k]^{-1} \right) \right] + \\ + \frac{\partial}{\partial \mu_j^k} \log[\mathcal{N}(\boldsymbol{\mu}_j^k|\widetilde{\boldsymbol{m}}_j^k, \xi_j^k)],\end{aligned} \tag{162}$$

$$\frac{\partial \mathfrak{L}(X_k, Z_k|Z_1 \dots Z_{k-1})}{\partial \Lambda_j^k} = \sum_{i=1}^{n_k} \frac{\partial}{\partial \Lambda_j^k} \left[\hat{p}(x_j^k|\boldsymbol{z}_i^k) \log \mathcal{N}\left(z_i^k \middle| \boldsymbol{\mu}_j^k, [\Lambda_j^k]^{-1} \right) \right] + \tag{163}$$

$$+\frac{\partial}{\partial \Lambda_j^k} \log\left[\mathcal{W}\left(\Lambda_j^k \middle| W_j^k, \nu_j^k\right)\right],$$

$$\frac{\partial \mathfrak{L}(X_k, Z_k | Z_1 \dots Z_{k-1})}{\partial \pi_j^k} = \frac{\partial}{\partial \pi_j^k}\left[\sum_{i=1}^{n_k} \hat{p}\left(x_j^k \middle| \boldsymbol{z}_i^k\right) \log \pi_j^k\right] - \lambda. \tag{164}$$

To analytically optimise this function, we need to meet the following extremum conditions:

$$\frac{\partial \log p(X_k, Z_k | Z_1 \dots Z_{k-1})}{\partial \boldsymbol{\mu}_j^k} = \mathbf{0}, \tag{165}$$

$$\frac{\partial \log p(X_k, Z_k | Z_1 \dots Z_{k-1})}{\partial \Lambda_j^k} = 0, \tag{166}$$

$$\frac{\partial \log p(X_k, Z_k | Z_1 \dots Z_{k-1})}{\partial \pi_j^k} = 0. \tag{167}$$

Then the normal and Wishart derivatives are estimated and substituted into the equation:

$$\frac{\partial}{\partial \boldsymbol{\mu}_j^k}\left\{\log\left[\mathcal{N}\left(\boldsymbol{z}_i^k \middle| \boldsymbol{\mu}_j^k, \left[\Lambda_j^k\right]^{-1}\right)\right]\right\} = -\frac{1}{2}\Lambda_j^k\left(\boldsymbol{\mu}_j^k - \boldsymbol{z}_i^k\right); \tag{168}$$

$$\begin{gathered}\frac{\partial}{\partial \Lambda_j^k}\left\{\log\left[\mathcal{N}\left(\boldsymbol{z}_i^k \middle| \boldsymbol{\mu}_j^k, \left[\Lambda_j^k\right]^{-1}\right)\right]\right\} = \\ = -\frac{1}{2}\frac{\partial}{\partial \Lambda_j^k}\left\{\log\left|\Lambda_j^k\right|\right\} - \frac{1}{2}\frac{\partial}{\partial \Lambda_j^k}\left[\left(\boldsymbol{z}_i^k - \boldsymbol{\mu}_j^k\right)^T \Lambda_j^k \left(\boldsymbol{z}_i^k - \boldsymbol{\mu}_j^k\right)\right] = \\ = -\frac{1}{2}\left[\Lambda_j^k\right]^{-1} + \left(\boldsymbol{z}_i^k - \boldsymbol{\mu}_j^k\right)\left(\boldsymbol{z}_i^k - \boldsymbol{\mu}_j^k\right)^T.\end{gathered} \tag{169}$$

$$\begin{gathered}\frac{\partial}{\partial \Lambda_j^k}\left\{\log\left[\mathcal{W}\left(\Lambda_j^k \middle| W_j^k, \nu_j^k\right)\right]\right\} = \\ = \frac{\partial}{\partial \Lambda_j^k}\left\{\frac{\left(\nu_j^k - l - 1\right)}{2}\log\left|\Lambda_j^k\right| - \frac{1}{2}\operatorname{tr}\left(\left[\mathrm{W}_j^k\right]^{-1}\Lambda_j^k\right)\right\} = \\ = \frac{\left(\nu_j^k - l - 1\right)}{2}\left[\Lambda_j^k\right]^{-1} - \frac{1}{2}\left[W_j^k\right]^{-1}.\end{gathered} \tag{170}$$

$$\frac{\partial}{\partial \boldsymbol{\mu}_j^k}\left\{\log\left[\mathcal{N}\left(\boldsymbol{\mu}_j^k \middle| \widetilde{\boldsymbol{m}}_j^k, \tilde{\xi}_j^k\right)\right]\right\} = -\frac{1}{2}\left[\tilde{\xi}_j^k\right]^{-1}\left(\boldsymbol{\mu}_j^k - \widetilde{\boldsymbol{m}}_j^k\right). \tag{171}$$

Here l is the dimensionality of the visible variables' feature space.

The final equations for the M-step after the solution of the equations (165), (166), (167) are calculated as follows:

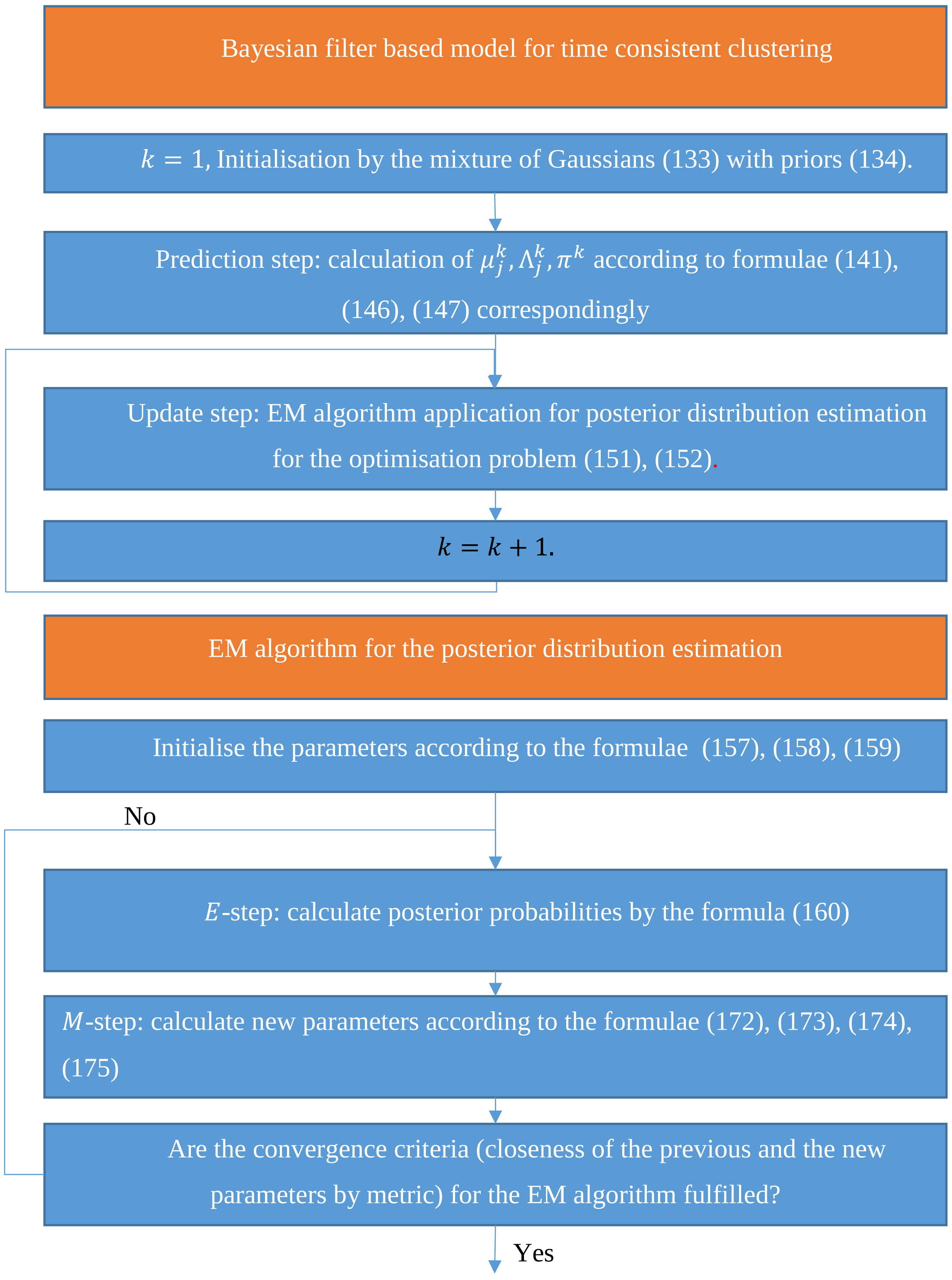

Bayesian filter based model for time consistent clustering
$k = 1$, Initialisation by the mixture of Gaussians (133) with priors (134).
Prediction step: calculation of $\mu_j^k, \Lambda_j^k, \pi^k$ according to formulae (141), (146), (147) correspondingly
Update step: EM algorithm application for posterior distribution estimation for the optimisation problem (151), (152).
$k = k + 1.$
EM algorithm for the posterior distribution estimation
Initialise the parameters according to the formulae (157), (158), (159)
No
E-step: calculate posterior probabilities by the formula (160)
M-step: calculate new parameters according to the formulae (172), (173), (174), (175)
Are the convergence criteria (closeness of the previous and the new parameters by metric) for the EM algorithm fulfilled?
Yes

Figure 7 Bayesian recursion for the proposed tracking algorithm

$$\boldsymbol{\mu}_j^k = \left\{\hat{p}\left(x_j^k \middle| \mathbf{z}_i^k\right)\left[\sum_{i=1}^{n_k} \Lambda_j^k\right] + \left[\xi_j^k\right]^{-1}\right\}^{-1} \left\{\hat{p}\left(x_j^k \middle| \mathbf{z}_i^k\right)\left[\sum_{i=1}^{n_k} \Lambda_j^k z_i^k\right] + \left[\xi_j^k\right]^{-1} \widetilde{\boldsymbol{m}}_j^k\right\}, \tag{172}$$

$$\left[\Lambda_j^k\right]^{-1} = \frac{\left(\left[\mathrm{W}_j^k\right]^{-1} - 2\sum_{i=1}^{n_k} \hat{p}\left(x_j^k \middle| \mathbf{z}_i^k\right)\left(\mathbf{z}_i^k - \boldsymbol{\mu}_j^k\right)\left(\mathbf{z}_i^k - \boldsymbol{\mu}_j^k\right)^T\right)}{n_j + v_j^k - l - 1}, \tag{173}$$

$$\pi_j^k = \frac{n_j^k}{\sum_{i=1}^{K} n_i^k}. \tag{174}$$

$$n_j^k = \sum_{i=1}^{n_k} \hat{p}\left(x_j^k \middle| \mathbf{z}_i^k\right). \tag{175}$$

The derived algorithm is summarised in Figure 7.

3.3 Bayesian filter based algorithm with variational inference

An alternative Bayesian filter implementation, proposed in section 3.2, uses variational approximate inference, adapted from the method [154] instead of the EM algorithm with subsequent Laplacian approximation. In the proposed method, the posterior probability is approximated by the mixture of Gaussians using variational inference.

The description starts from the variational inference algorithm description, which was adapted from [154]. After that its incorporation into the Bayesian filter is shown, with the reasoning to separate the original Bayesian filter model from the previously known variational inference approximation for Gaussian mixtures.

3.3.1 Variational inference for the Bayesian filter approximation

The variational inference model derivation is described using the model from [154], with the difference that in this research the original parameters for the clusters are non-symmetric as they should use the information from the prediction step.

First, the general variational approximation approach based on the likelihood optimisation is considered [154]. Starting from the joint distribution $p(Z, X)$, where Z are visible variables, and X are latent, the aim is to approximate the posterior distribution $p(X|Z)$ in the factorised form

$$\tilde{p}(X) = \prod_{i=1}^{N} \tilde{p}_i(X_i), \tag{176}$$

where $X = \{X_1 \dots X_N\}$, using the lower bound maximisation criteria (see section 2.2.6.1):

$$L(\tilde{p}) = \int \prod_{i=1}^{N} \tilde{p}_i \ln \frac{p(Z,X)}{\prod_{i=1}^{N} \tilde{p}_i}\, dX \to \max_{\tilde{p}}, \tag{177}$$

where $\tilde{p}_i, \tilde{p}$ are short notations for $\tilde{p}_i(X_i), \tilde{p}(X)$.

Given fixed $\tilde{p}_i \; \forall\, i = [1 \ldots N] \setminus j, j \in [1 \ldots N]$, the following expressions can be written:

$$L(\tilde{p}) = \int \prod_{i=1}^{N} \tilde{p}_i \left(\ln p(Z,X) - \sum_{i=1}^{N} \ln \tilde{p}_i \right) dX =$$
$$= \int \tilde{p}_j \left(\ln p(Z,X) \prod_{\substack{i=1,\\ i\neq j}}^{N} dX_j \right) - \int \tilde{p}_j \ln \tilde{p}_j \, dX_j + \text{const.} \tag{178}$$

Then the functional $L(\tilde{p})$ can be maximised with respect to all possible forms of $\tilde{p}_i(X_i)$. One can notice that this formula represents a negative Kullback-Leibler divergence. Hence the optimisation problem (177) can be proven to be a Kulback-Leibler divergence minimisation problem, which gives the following solution:

$$\ln \tilde{p}_j(X_i) = \mathbb{E}_{i\neq j} \ln p(Z,X) + \text{const.} \tag{179}$$

Here, the constant term is determined by the probability normalisation condition. Finally one can obtain

$$\mathbb{E}_{i\neq j} \ln p(Z,X) = \int \ln p(Z,X) \prod_{\substack{i=1,\\ i\neq j}}^{N} dX_j. \tag{180}$$

This general result can be used to obtain the particular solution for the stated variational approximation problem described below. Define a set of visible variables $\boldsymbol{Z}_k = \{\boldsymbol{z}_1^k, \boldsymbol{z}_2^k \ldots \boldsymbol{z}_{n_k}^k\}$ with the following parameters of the Gaussian mixtures: means $\boldsymbol{\mu}_k = \{\boldsymbol{\mu}_1^k, \boldsymbol{\mu}_2^k \ldots \boldsymbol{\mu}_K^k\}$, covariance matrices $\Sigma_k = \{\Sigma_1^k, \Sigma_2^k \ldots \Sigma_K^k\}$, and weights $\pi_k = \{\pi_1^k, \pi_2^k \ldots \pi_K^k\}$. The belonging to each of the Gaussians is described by the K-dimensional vectors of the weights $\{0, 1\}$ for each of the Gaussians, $\boldsymbol{V}_k = \{\boldsymbol{v}_1^k, \boldsymbol{v}_2^k \ldots \boldsymbol{v}_{n_k}^k\}$. All these variables $\{\boldsymbol{\mu}_k, \Sigma_k, \pi_k, \boldsymbol{V}_k\} = X_k$ are considered as hidden variables altogether. The time index k for the Gaussian mixture is omitted further in this section to make the variational approximation concept clearer.

The overall model for the Gaussian mixture is described by the following joint distribution:

$$p(\boldsymbol{Z}, \boldsymbol{V}, \pi, \boldsymbol{\mu}, \Lambda) = p(\boldsymbol{Z}|\boldsymbol{V}, \boldsymbol{\mu}, \Lambda) p(\boldsymbol{V}|\pi) p(\pi) p(\boldsymbol{\mu}|\Lambda) p(\Lambda), \tag{181}$$

where

$$p(\boldsymbol{Z}|\boldsymbol{V},\boldsymbol{\mu},\Lambda) = \prod_{i=1}^{n}\prod_{j=1}^{K}\mathcal{N}\left(\boldsymbol{z}_n\middle|\boldsymbol{\mu}_j,\Lambda_j^{-1}\right), \tag{182}$$

$$p(\boldsymbol{V}|\pi) = \prod_{i=1}^{n}\prod_{j=1}^{K}\pi_j^{x_{1j}}, \tag{183}$$

$$p(\pi) = \text{Dir}\,(\pi|\alpha^0) = C(\alpha^0)\prod_{j=1}^{K}\pi_j^{\alpha_k^0-1}, \tag{184}$$

and $\alpha^0 = \begin{pmatrix}\alpha_1^0\\ \dots\\ \alpha_K^0\end{pmatrix}$ is the parameters vector for the Dirichlet distribution.

$$p(\boldsymbol{\mu},\Lambda) = \prod_{j=1}^{K}\mathcal{N}\left(\boldsymbol{\mu}_j\middle|\boldsymbol{m}_j^0,\left(\beta_j^0\Lambda_j\right)^{-1}\right)\mathcal{W}\left(\Lambda_j\middle|W_j^0,\nu_j^0\right), \tag{185}$$

where $\boldsymbol{m}_j^0,\beta_j^0,W_j^0,\nu_j^0$ are the initial parameters of the distributions. Here, following [154], the following 1-of-K notation is used:

$$\boldsymbol{v}_{ik} = \begin{cases}1, \boldsymbol{z}_i \text{ was sampled from the } k-t\text{h Gausian,}\\ \qquad\qquad\qquad 0, \text{else.}\end{cases} \tag{186}$$

The analytical forms of the distributions on the parameters are determined by the requirements for the conjugate priors.

The aim is to approximate the posterior distribution $p(\boldsymbol{Z},\pi,\mu,\Lambda)$ by factorisation onto the parameters probabilities and the posterior probability in the following way:

$$\tilde{p}(\boldsymbol{V},\pi,\boldsymbol{\mu},\Lambda) = \tilde{p}(\boldsymbol{V})\tilde{p}(\pi,\boldsymbol{\mu},\Lambda). \tag{187}$$

Then the factors $\tilde{p}(\boldsymbol{V})$ and $\tilde{p}(\pi,\boldsymbol{\mu},\Lambda)$ can be estimated

$$\ln\tilde{p}(\boldsymbol{V}) = \mathbb{E}_{\pi,\mu,\Lambda}\ln p(\boldsymbol{Z},\boldsymbol{V},\pi,\boldsymbol{\mu},\Lambda) + \text{const}, \tag{188}$$

$$\ln\tilde{p}(\pi,\boldsymbol{\mu},\Lambda) = \mathbb{E}_X\ln p(\boldsymbol{Z},\boldsymbol{V},\pi,\boldsymbol{\mu},\Lambda) + \text{const}. \tag{189}$$

Let us first consider the expression (188). One can notice that

$$\ln\tilde{p}(\boldsymbol{V}) = \mathbb{E}_{\boldsymbol{\mu},\Lambda}\ln p(\boldsymbol{Z}|\boldsymbol{V},\mu,\Lambda) + \mathbb{E}_\pi\ln p(\boldsymbol{V}|\pi) + \text{const}. \tag{190}$$

Then both the terms can be considered separately:

$$\mathbb{E}_\pi\ln p(\boldsymbol{Z}|\boldsymbol{V},\mu,\Lambda\,) = \mathbb{E}_\pi\ln\prod_{i=1}^{N}\prod_{k=1}^{K}\pi_k^{v_{ik}} = \sum_{i=1}^{N}\sum_{k=1}^{K}\boldsymbol{v}_{ik}\mathbb{E}_{\pi_k}[\ln\pi_k] \tag{191}$$

and

$$\mathbb{E}_{\mu,\Lambda}\ln p(Z|\boldsymbol{V},\mu,\Lambda) = \mathbb{E}_{\mu,\Lambda}\ln\prod_{i=1}^{N}\prod_{k=1}^{K}[\mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k,\Lambda_k^{-1})]^{\boldsymbol{w}_{ik}} =$$

$$= \sum_{i=1}^{N}\sum_{k=1}^{K}\boldsymbol{x}_{ik}\mathbb{E}_{\mu_k,\Lambda_\mathrm{k}}\ln[\mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k,\Lambda_k^{-1})] = \tag{192}$$

$$= \sum_{\substack{i=1,\\k=1}}^{N,K}\boldsymbol{x}_{ik}\left(-\frac{l}{2}\ln 2\pi + \frac{1}{2}\mathbb{E}_{\Lambda_\mathrm{k}}\ln\det\Lambda_k - \frac{1}{2}\mathbb{E}_{\boldsymbol{\mu}_k,\Lambda_\mathrm{k}}[(\boldsymbol{x}_i-\boldsymbol{\mu}_k)^T\Lambda_k^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_k)]\right),$$

where, as in the previous sections, l is the dimensionality of the covariance matrices. After combining summands one can obtain the following equation:

$$\ln\tilde{p}(\boldsymbol{V}) = \prod_{i=1}^{N}\prod_{k=1}^{K}r_{ik}{}^{\boldsymbol{w}_{ik}}, \tag{193}$$

where

$$\ln r_{ik} = -\frac{l}{2}\ln[2\pi] + \frac{1}{2}\mathbb{E}_{\Lambda_\mathrm{k}}\ln\det\Lambda_k -$$
$$-\frac{1}{2}\mathbb{E}_{\mu_k,\Lambda_\mathrm{k}}[(\boldsymbol{z}_i-\boldsymbol{\mu}_k)^T\Lambda_k^{-1}(\boldsymbol{z}_i-\boldsymbol{\mu}_k)] + \mathbb{E}_{\pi_k}[\ln\pi_k]. \tag{194}$$

Then, using the normalisation condition, one can obtain

$$\tilde{p}(\boldsymbol{V}) = \prod_{i=1}^{N}\prod_{k=1}^{K}\gamma_{ik}{}^{v_{ik}}, \tag{195}$$

where

$$\gamma_{ik} = \frac{r_{ik}}{\sum_{j=1}^{K}r_{ij}}. \tag{196}$$

Formula (189) can be represented as

$$\ln\tilde{p}(\pi,\boldsymbol{\mu},\Lambda) = \mathbb{E}_{\boldsymbol{V}}\ln p(\boldsymbol{Z},\boldsymbol{V},\pi,\boldsymbol{\mu},\Lambda) + \mathrm{const} =$$
$$= \mathbb{E}_{\boldsymbol{V}}[\ln p(\boldsymbol{Z}|\boldsymbol{V},\mu,\Lambda) + \ln p(\boldsymbol{V}|\pi)] + \ln p(\pi) + \ln[p(\boldsymbol{\mu}|\Lambda)p(\Lambda)] + \mathrm{const} =$$
$$= \sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{E}_{\boldsymbol{V}}[v_{ik}]\ln[\mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k,\Lambda_k^{-1})] + \mathbb{E}_{\boldsymbol{V}}\ln p(\boldsymbol{V}|\pi) + \tag{197}$$
$$+\ln p(\pi) + \ln p(\boldsymbol{\mu}|\Lambda) + \ln p(\Lambda) + \mathrm{const}.$$

One can see that

$$\tilde{p}(\pi,\boldsymbol{\mu},\Lambda) = \tilde{p}(\pi)\prod_{k=1}^{K}\tilde{p}(\boldsymbol{\mu}_k,\Lambda_\mathrm{k}). \tag{198}$$

Then, after considering the components of this product, one can see that

$$\tilde{p}(\pi) = \mathbb{E}_X \ln p(\boldsymbol{V}|\pi) + \ln p(\pi) + \text{const}, \tag{199}$$

$$\begin{gathered}\tilde{p}(\,\boldsymbol{\mu}, \Lambda) = \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{V}}[w_{ik}]\ln[\mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k, \Lambda_k^{-1})] + \\ + \sum_{k=1}^{K} \ln p(\boldsymbol{\mu}_k|\Lambda_k) + \sum_{k=1}^{K} \ln p(\Lambda_\mathrm{k}) + \text{const.}\end{gathered} \tag{200}$$

Then, consider these approximated distributions and obtain

$$\mathbb{E}_{\boldsymbol{V}} \ln p(\boldsymbol{V}|\pi) = \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{V}}[w_{ik}] \ln \pi_k = \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{ik} \ln \pi_k. \tag{201}$$

$$\ln p\ (\pi) = \ln C(\boldsymbol{\alpha}^0) + \sum_{k=1}^{K} (\boldsymbol{\alpha}_k^0 - 1) \ln \pi_k, \tag{202}$$

where $\boldsymbol{\alpha}^0 = \begin{pmatrix} \alpha_1^0 \\ \dots \\ \alpha_K^0 \end{pmatrix}$ are the parameters of the Dirichlet distribution.

After substitution, the resulting equations for $\tilde{p}(\pi)$ are given as:

$$\begin{gathered}\tilde{p}(\pi) = \ln C(\boldsymbol{\alpha}_0) + \sum_{k=1}^{K} \ln \pi_k \left((\boldsymbol{\alpha}_0 - 1) + \sum_{i=1}^{N} \gamma_{ik} \right) + \text{const} = \\ = \ln C(\boldsymbol{\alpha}_0) + \sum_{k=1}^{K} \ln \pi_k \left((\boldsymbol{\alpha}_0 - 1) + n_k \right) + \text{const},\end{gathered} \tag{203}$$

where

$$n_k = \sum_{i=1}^{N} \gamma_{ik}. \tag{204}$$

Then, one can see that after normalisation

$$\tilde{p}(\pi) = \text{Dir}\ (\pi|\boldsymbol{\alpha}), \tag{205}$$

where $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \dots \\ \alpha_K \end{pmatrix} = \begin{pmatrix} \alpha_1^0 + n_k \\ \dots \\ \alpha_K^0 + n_k \end{pmatrix}$.

Then we calculate

$$\tilde{p}(\boldsymbol{\mu}, \Lambda) = \prod_{j=1}^{K} \mathcal{N}\left(\boldsymbol{\mu}_j \middle| \boldsymbol{m}_j, (\beta_j \Lambda_j)^{-1}\right) \mathcal{W}(\Lambda_j | W_j, \nu_j). \tag{206}$$

$$\begin{gathered}\tilde{p}(\boldsymbol{\mu},\Lambda) = \\ = \sum_{i=1}^{N}\sum_{k=1}^{K}\gamma_{ik}\ln[\mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k,\Lambda_k^{-1})] + \sum_{k=1}^{K}\ln p(\boldsymbol{\mu}_k|\Lambda_k) + \sum_{k=1}^{K}\ln p(\Lambda_\mathrm{k}) + \mathrm{const} = \\ = \sum_{k=1}^{K}\left[\sum_{i=1}^{N}\gamma_{ik}\ln\mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_k,\Lambda_k^{-1}) + \ln\mathcal{N}\left(\boldsymbol{\mu}_k\middle|\boldsymbol{m}_k^0,\frac{{\Lambda_k}^{-1}}{\beta_k^0}\right)\mathcal{W}(\Lambda_k|W_k^0,\nu_k^0)\right] + \\ +\mathrm{const}.\end{gathered} \tag{207}$$

It can be proven that [154]

$$\tilde{p}(\boldsymbol{\mu},\Lambda) = \mathcal{N}(\boldsymbol{\mu}_k|\boldsymbol{m}_k,(\beta_k\Lambda_k)^{-1})\mathcal{W}(\Lambda_k|W_k,\nu_k), \tag{208}$$

where

$$\beta_k = \beta_k^0 + n_k, \tag{209}$$

$$\boldsymbol{m}_k = \frac{(\beta_k^0\boldsymbol{m}_k^0 + n_k\bar{\boldsymbol{z}}_k)}{\beta_k}, \tag{210}$$

$$\bar{\boldsymbol{z}}_k = \frac{\sum_{i=1}^{N}\gamma_{ik}\boldsymbol{z}_i}{n_k}, \tag{211}$$

$$W_k^{-1} = [W_k^0]^{-1} + n_k\bar{\Sigma}_k + \frac{\beta_k^0 n_k}{\beta_k^0 + n_k}(\bar{\boldsymbol{z}}_k - \boldsymbol{m}_k^0)(\bar{\boldsymbol{z}}_k - \boldsymbol{m}_k^0)^T, \tag{212}$$

$$\nu_k = \nu_k^0 + \gamma_k + 1. \tag{213}$$

One can see that there is an interdependence between the γ_{ik} weights for each of the distributions and the parameters of the Gaussian mixture [154]. Therefore, we should find $\ln r_{ik}$ and then γ_{ik} by the normalisation of r_{ik} [154]:

$$\begin{gathered}\ln r_{ik} = -\frac{l}{2}\ln[2\pi] + \frac{1}{2}\mathbb{E}_{\Lambda_\mathrm{k}}\ln\det\Lambda_k - \\ -\frac{1}{2}\mathbb{E}_{\boldsymbol{\mu}_k,\Lambda_\mathrm{k}}[(\boldsymbol{z}_i-\boldsymbol{\mu}_k)^T\Lambda_k^{-1}(\boldsymbol{z}_i-\boldsymbol{\mu}_k)] + \mathbb{E}_{\pi_k}[\ln\pi_k],\end{gathered} \tag{214}$$

$$\mathbb{E}_{\Lambda_\mathrm{k}}\ln\det\Lambda_k = \sum_{i=1}^{l}\psi\left(\frac{\nu_k+1-i}{2}\right) + l\ln 2 + \ln\det W_k, \tag{215}$$

$$\mathbb{E}_{\mu_k,\Lambda_\mathrm{k}}[(\boldsymbol{x}_i-\boldsymbol{\mu}_k)^T\Lambda_k^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_k)] = l\beta_k^{-1} + \nu_k(\boldsymbol{x}_i-\boldsymbol{\mu}_k)^T W_k(\boldsymbol{x}_i-\boldsymbol{\mu}_k), \tag{216}$$

$$\mathbb{E}_{\pi_k}[\ln\pi_k] = \psi(\alpha_k) - \psi\left(\sum_{i=1}^{K}\alpha_k\right), \tag{217}$$

where ψ is the digamma function defined as

$$\psi(\alpha) = \frac{\mathrm{d} \ln \Gamma(\alpha)}{\mathrm{d}\alpha}, \tag{218}$$

$\Gamma(\alpha)$ is the gamma function [194], which is an extension of factorial function $(\alpha - 1)!$ to real and complex numbers.

Substituting it into the equation for r_{ik}, we finally obtain:

$$\ln r_{ik} = -\frac{l}{2}\ln[2\pi] + \frac{1}{2}\left(\sum_{i=1}^{l} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + l \ln 2 + \ln \det W_k\right) - \\ -\frac{1}{2}\left(l\beta_k^{-1} + \nu_k(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T W_k(\boldsymbol{x}_i - \boldsymbol{\mu}_k)\right) + \psi(\alpha_k) - \psi\left(\sum_{i=1}^{K} \alpha_k\right) + \text{const.} \tag{219}$$

Using these formulae, one can formulate a procedure, resembling EM algorithm, capable of iterative optimisation of the maximum likelihood (see Figure 8).

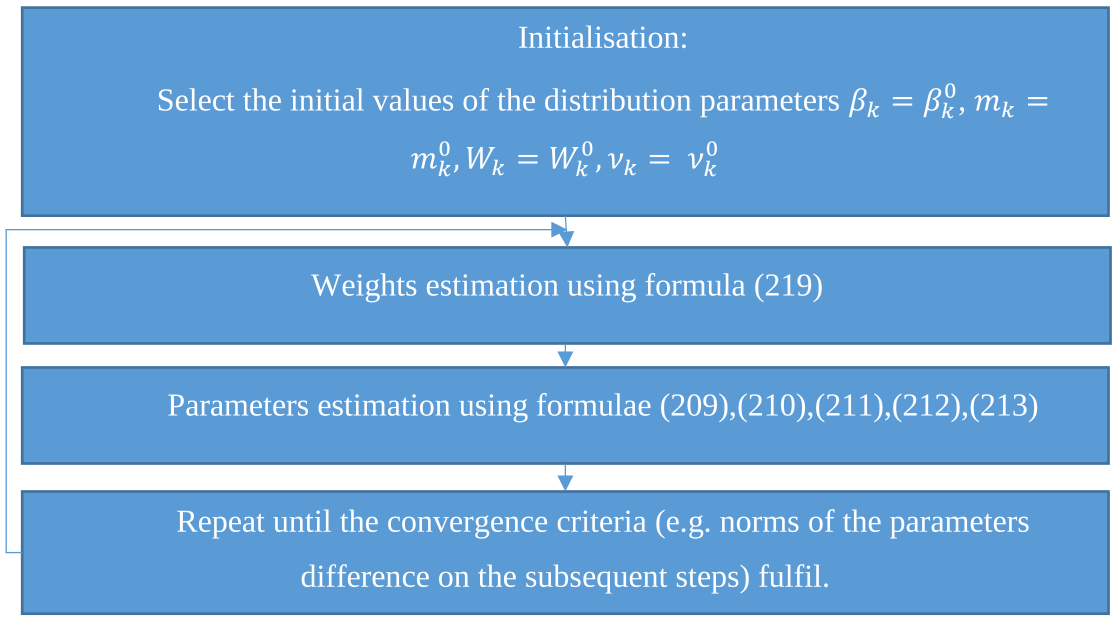

Initialisation:
Select the initial values of the distribution parameters $\beta_k = \beta_k^0, m_k = m_k^0, W_k = W_k^0, \nu_k = \nu_k^0$
Weights estimation using formula (219)
Parameters estimation using formulae (209),(210),(211),(212),(213)
Repeat until the convergence criteria (e.g. norms of the parameters difference on the subsequent steps) fulfil.

Figure 8 The scheme of the variational approximation algorithm based on [154]

3.4 Feature points detection and tracking

In order to define the video tracking algorithm, it is necessary to define the feature space which is used in the video analysis algorithm. For this purpose, the following procedure is proposed:

- detect the fixed number of object points from the image using non-maximum suppression on the Harris corner map [191];
- track them using the Lucas-Kanade algorithm [56], for those which cannot be reliably tracked, search for the replacement using the Harris corner map;

- for each of the points p, which has been tracked for N_T frames, apply the Kabsch algorithm [155] between the current frame and the pre-defined number of frames N_T before as it is described in section 3.2.2 and calculate the velocities from it;
- for each of the points, which has been tracked for N_T frames, create the feature vector $\boldsymbol{z} = \{x, y, v_x, v_y\}$, where x and y are the point's screen co-ordinates, and v_x and v_y are the point's velocity projections for x and y respectively.

The vector $\boldsymbol{z}$ is used as a visible variable within both implementations of the Bayesian filters.

3.5 Object detection

The objects and the background are distinguished using the following procedure. First, the rotation matrix and translation vector are estimated using Kabsch algorithm [155] (see section 3.2.2). Then the following heuristic procedure is used for the threshold estimation [15]. First, all points $\boldsymbol{z}_i$ are sorted by their Kabsch algorithm L^2 errors $\left|\boldsymbol{z}_i^k - R\boldsymbol{z}_i^{k-1} - T\right|_{L^2}$, then the standard deviation S for the Kabsch algorithm errors is estimated. The threshold is defined as a mean between the neighbouring (in sense of L^2 metric) velocity vectors, for which the difference is more than τS, where τ is some pre-defined constant (normally $3 \leq \tau \leq 20$). To exclude ambiguous thresholds, such points with the least errors according to L^2 metric are selected (see Figure 9).

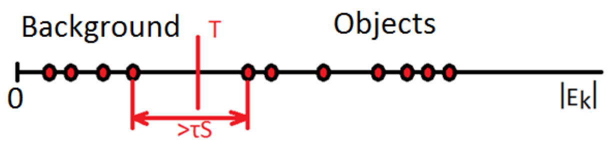


Figure 9. Data thresholding [15]. The red points on the line are the points from G_k.

Then, each cluster, obtained after the tracking stage, is treated by the thresholding with some pre-defined threshold T. Then, all the clusters having more than half of the points with a speed, larger than the threshold T, are selected as the clusters belonging to the OoI.

3.6 Object co-ordinates estimation combined with Bayesian filter based algorithm

The next problem solved during this research is the combination of the distance estimation method with the proposed Bayesian filter implementation [15]. Here, the proposed geographical co-ordinates estimation approach is described [15]. It uses the fusion between the video footage and the synchronised inertial sensors data.

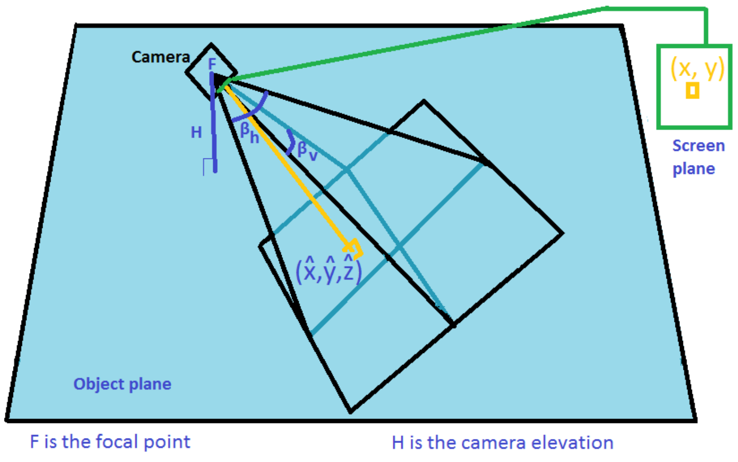


Figure 10. The geometrical description for the proposed method [15]

In the scheme, depicted in figure 10, one can see the surface which is assumed to be ideal horizontal plane. At each moment, the elevation of the camera above the object plane that is above the object, as well as a screen plane inclination, fully described by Euler angles [156], and geographical co-ordinates are known from the inertial sensor measurements. The central projection model is used for the camera to estimate the distances to the object given all these assumptions.

Then, the following algorithm is proposed for the distance and direction estimation for the vector from the camera to the object, as well as real-world co-ordinates (latitude and longitude) object position. Given the point (x, y) in the video frame I_k with sizes $w \times h$, one can obtain centred screen point position

$$\hat{x} = x - \frac{w}{2}, \hat{y} = \frac{h}{2} - y. \tag{220}$$

Then one can estimate the camera direction in the globally tethered North-East-Down (NED) co-ordinate system:

$$\boldsymbol{n} = (\tilde{x}, \tilde{y}, \tilde{z}) = \mathcal{R}\begin{pmatrix} \frac{2\tan\left(\frac{\beta_h}{2}\right)\hat{x}}{h} & \frac{2\tan\left(\frac{\beta_v}{2}\right)\hat{y}}{w} & 1 \end{pmatrix}^T, \tag{221}$$

where $\mathcal{R}$ is the world rotation matrix composed from the Euler angles [156], and β_h, β_v are the horizontal and vertical angles of view.

The distance to the object is determined as $\left|\frac{\mathrm{H}}{\tilde{z}}n\right|$, and, after the re-scaling, the Vincenty formulae [158] can be applied to estimate the real-world co-ordinates. The algorithm is summarised in Figure 11.

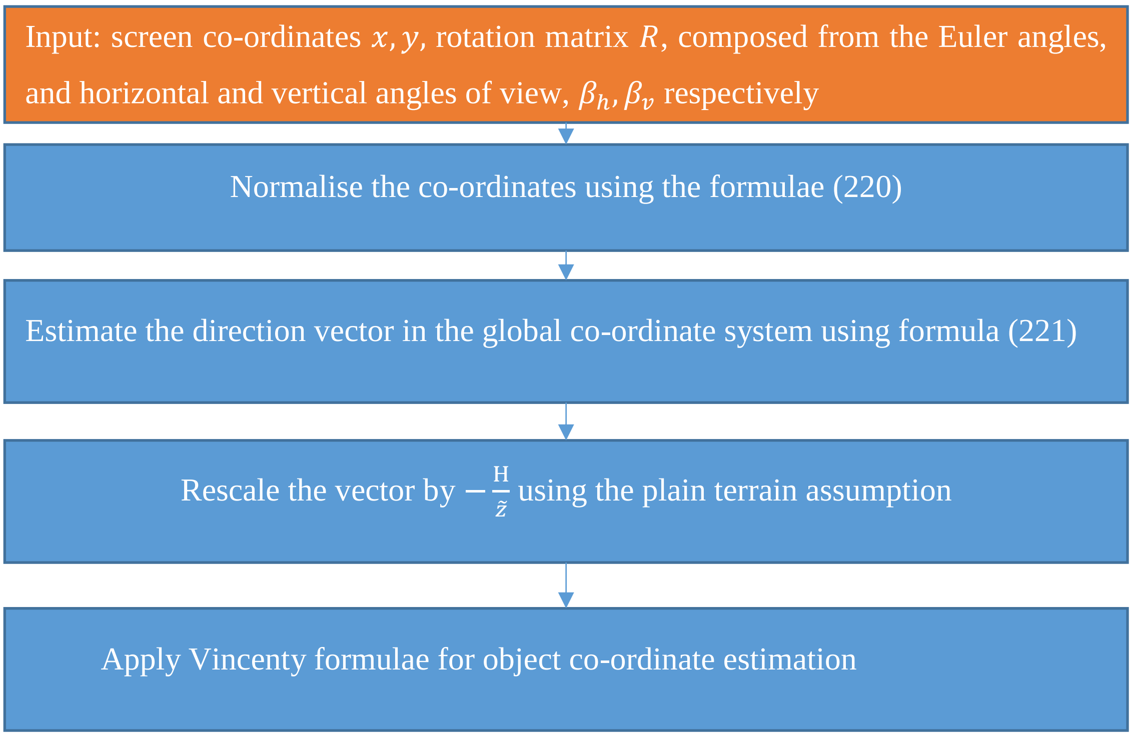

Input: screen co-ordinates x, y, rotation matrix R, composed from the Euler angles, and horizontal and vertical angles of view, β_h, β_v respectively
Normalise the co-ordinates using the formulae (220)
Estimate the direction vector in the global co-ordinate system using formula (221)
Rescale the vector by $-\frac{\mathrm{H}}{\tilde{z}}$ using the plain terrain assumption
Apply Vincenty formulae for object co-ordinate estimation

Figure 11 Geo-position estimation, assuming the plain terrain

3.7 Final formulation of the proposed tracking algorithm

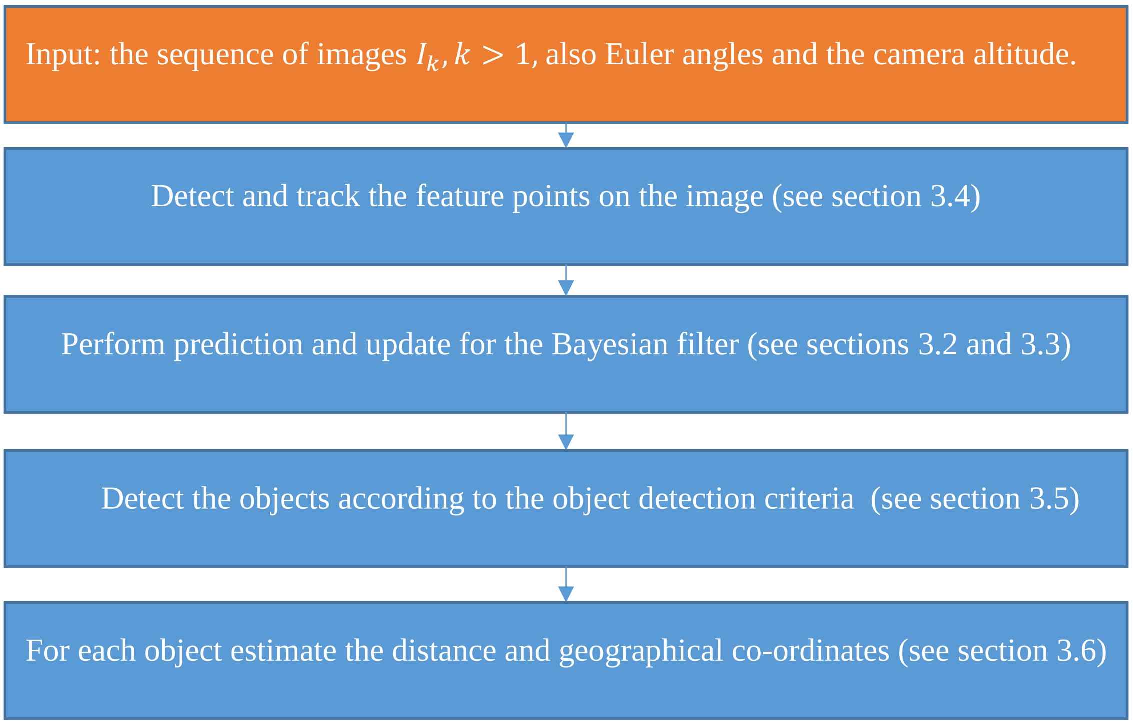

Input: the sequence of images $I_k, k > 1$, also Euler angles and the camera altitude.
Detect and track the feature points on the image (see section 3.4)
Perform prediction and update for the Bayesian filter (see sections 3.2 and 3.3)
Detect the objects according to the object detection criteria (see section 3.5)
For each object estimate the distance and geographical co-ordinates (see section 3.6)

Figure 12 Summary of the proposed tracking algorithm

The assumption about the possibility of application of the proposed time-consistent Gaussian mixtures clustering method to the video object tracking problem needs to be discussed. More precisely, the correctness of the Gaussian mixtures model application is justified by the following practical considerations:

- it is assumed that each object generates a number of points (in a joint position and velocity space) that are close enough and concentrated near the object centre that can be approximated by a Gaussian distribution,
- feature points, in a joint position and velocity space, are scattered densely near objects (either background or moving objects), and sparsely in between that allows to build up a mixture model, and
- while other density models could be considered, e.g. based on particle filters, the advantage of Gaussian model is in its computational simplicity that paves the way for real time applications.

The experiments, given further in this thesis in section 5.1, show the evidence of viability of the assumptions.

The final algorithm is obtained as a combination of the methods, described in the previous sections, and is shown in Figure 12. From these stages one can see that the algorithm is actually composed from three parts:

a) a part, specific for the video (detection and tracking of the feature points, object detection based on pre-defined criteria);
b) a general part which contains Bayesian filtering;
c) the distance estimation and geographical co-ordinates estimation.

Therefore, it is clearly visible that the general and domain-specific parts, related to video and geographical positioning, can be separated from each other.

3.8 Conclusion

One of the most critical problems of video surveillance is tracking. In this section, two versions of the algorithm have been proposed, both relying on the concept of time-consistent clustering and implementing it. The methods have been first formulated in sections 3.2 and 3.3, for Laplacian and variational update step approximations correspondingly. Their formulation is general that ensures the applications of the filter to problems far beyond the scope of video analytics, in a time-consistent clustering of the dataset, evolving in time. The Laplacian update step approximation is based on the well-known EM algorithm, which has been reviewed in section 2.2.6.1. The variational update step is based on the variational approximation [154].

The rest of the chapter describes an application of the method to the domain of video analytics, showing its applicability for real time moving objects detection for a moving camera. However, the detection can be alternatively implemented using classification and clustering methods, including those presented in chapter 4. The object co-ordinates estimation algorithm for the plain terrain assumption is presented in section 3.6. The chapter is finished with the formulation of the proposed algorithm for object detection and tracking, combining the proposed methods.

4 Proposed object detection and recognition techniques

The object tracking, detection and recognition models are often a part of the solution of the wider problem of image understanding. Therefore, tracking models, like those described in chapter 3, can be combined with object detection and recognition models, which are described in this section.

Nowadays, there are plenty of algorithms for object detection and recognition; some of them, which are relevant to the proposed methods, are briefly described in section 2.2. As it was shown in the state-of-the-art review, these methods are based on diverse techniques. In this section, we describe those of them that fulfil several special criteria, which can be required for many practical problems [19]:

- *ability of incremental learning*: the classifier should not repeat all the whole procedure of the classifier learning or parameters adjustment when adding new sample vectors into the training set;
- *online design*: the system should not require holding of all data in the memory when learning incrementally;
- *evolving structure*: the algorithm should be capable to enforce the new samples from the training set when learning incrementally and forget old and outlier samples.

The algorithms, described here, except Chan-Vese algorithm modifications, described in section 4.4, aim to fulfil these requirements and can be used for detection and recognition as a part of the object detection and tracking frameworks. The image segmentation method, described in section 4.4, uses a novel optimisation method for the Chan-Vese functional as well as it improves the functional itself, making it, in contrast to the original technique, non-parametric.

The rest of the chapter is organised as follows. The clustering techniques, based on TEDA, are described in section 4.1. The classification techniques, based on TEDA, are described in section 4.2. The SVM incremental training procedure with trainable kernels is described in section 4.3. The image segmentation algorithms, based on Chan-Vese functional modifications, are described in section 4.4.

4.1 Clustering techniques

This section reviews the clustering techniques based on the TEDA approach. First the brief review of the Typicality and Eccentricity Data Analysis (TEDA) framework proposed in [159] is carried out in section 4.1.1 as a basis for the proposed algorithms. After that, the recursive calculation of TEDA quantities for certain types of distances are described in section 4.1.2. The specific problem of recursive covariance matrix update is discussed in section 4.1.3. The TEDACluster algorithm is described in section 4.1.4, while the version of this algorithm for big data applications in described in section 4.5.1.

4.1.1 TEDA approach overview

The main idea of the TEDA framework [159] is to use the data typicality and eccentricity scores for the data set for the analysis of the statistical properties of data, using the ratios of between-point distances.

The TEDA approach is stated as follows [159]. Define a set of objects described by the features from some feature space $\mathfrak{X}$. For simplicity, the features $\boldsymbol{x} \in \mathfrak{X}$ for some particular object are referred as 'data samples'. Some between sample distance or similarity $d(\boldsymbol{x}, \boldsymbol{y}): \mathfrak{X} \to \mathbb{R}, \boldsymbol{x}, \boldsymbol{y} \in \mathfrak{X}$, is defined then within the feature space. Then one can have the data sample set $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k\}, \boldsymbol{x}_i \in \mathfrak{X}$. The sum distance to any particular data sample $\boldsymbol{y} \in \mathfrak{X}$ (inside or outside the data sample set S) is defined as [159]:

$$\pi^k(\boldsymbol{y}) = \sum_{i=1}^{k} d(\boldsymbol{y}, \boldsymbol{x}_i), k \geq 1. \tag{222}$$

The data sample eccentricity is defined as [159]:

$$\xi^k(\boldsymbol{y}) = \frac{2\pi^k(\boldsymbol{y})}{\sum_{i=1}^{k} \pi^k(\boldsymbol{y})} = 2\frac{\sum_{i=1}^{k} d(\boldsymbol{y}, \boldsymbol{x}_i)}{\sum_{i=1}^{k}\sum_{j=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x}_j)}. \tag{223}$$

Here "2" stands as the distance between $\boldsymbol{x}_i, \boldsymbol{x}_j$ is presented two times in the sum in the denominator, as $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and as $d(\boldsymbol{x}_j, \boldsymbol{x}_i)$. As a complement of the data eccentricity, a data typicality is defined as [159]:

$$\tau^k(\boldsymbol{y}) = 1 - \xi^k(\boldsymbol{y}). \tag{224}$$

In all these cases, the data typicality can be calculated recursively (with special simple equations for the case of Euclidean and Mahalanobis distances), therefore the index k stands in the notation for $\xi^k(\boldsymbol{y}), \tau^k(\boldsymbol{y}), \pi^k(\boldsymbol{y})$. Both the quantities, typicality and eccentricity, can be summed up to constant [159]:

$$\sum_{j=1}^{k} \xi^k(\boldsymbol{x}_j) = 2, \sum_{j=1}^{k} \tau^k(\boldsymbol{x}_j) = k - 2, \tag{225}$$

therefore, they can be normalised[159]:

$$\zeta^k(\boldsymbol{x}) = \frac{\xi^k(\boldsymbol{x})}{2}, \sum_{i=1}^{k} \zeta^k(\boldsymbol{x}_i) = 1, k \geq 2, \tag{226}$$

$$t^k(\boldsymbol{x}) = \frac{\tau^k(\boldsymbol{x})}{k-2}, \sum_{i=1}^{k} t^k(\boldsymbol{x}_i) = 1, k > 2. \tag{227}$$

The quantities above are referred to as normalised typicality and eccentricity, respectively.

4.1.2 Recursive calculation of typicality and eccentricity

Generally the incremental equations for TEDA are given as follows:

$$\pi^k(\boldsymbol{y}) = \pi^{k-1}(\boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{x}_k) \tag{228}$$

that can be easily checked from formula (222).

For some particular densities, like Euclidean and Mahalanobis, there exist more efficient update formulae than for the general case, helping to fulfil the evolving systems conditions described in the beginning of the chapter such as avoid the storage of all the data set and, thus, enable the online update. It turns out that for these distances TEDA scores can be expressed through mean, variance and covariance, each of which can be updated recursively [19].

4.1.2.1 *Euclidean case*

First, the case of Euclidean distance $d(\boldsymbol{x}, \boldsymbol{y})$ between two points $\boldsymbol{x}, \boldsymbol{y}$ in the data space $\mathfrak{X}$ is considered. After substituting Euclidean distance expressions, the eccentricity is described as [19]

$$\begin{aligned}
\xi^k(\boldsymbol{x}) &= 2\frac{\sum_{i=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{i=1}^{k}\sum_{j=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x}_j)} = 2\frac{\sum_{i=1}^{k} (\boldsymbol{x}_i - \boldsymbol{x})^T(\boldsymbol{x}_i - \boldsymbol{x})}{\sum_{i=1}^{k}\sum_{j=1}^{k} (\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j)} = \\
&= \left\{ \begin{array}{l} \boldsymbol{\mu}_x^k = \sum_{i=1}^{k} \frac{\boldsymbol{x}_i}{k}, \mu_{x^Tx} = \sum_{i=1}^{k} \frac{\boldsymbol{x}_i^T\boldsymbol{x}_i}{k}, \\ [\sigma_x^k]^2 = \sum_{i=1}^{k} \frac{(\boldsymbol{x}_i - \boldsymbol{\mu}_x^k)^T(\boldsymbol{x}_i - \boldsymbol{\mu}_x^k)}{k} \end{array} \right\} = \frac{2k\left(\mu_{x^Tx}^k - 2[\boldsymbol{\mu}_x^k]^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{x}\right)}{k^2\left(2\mu_{x^Tx}^k - 2{\boldsymbol{\mu}_x^k}^T\boldsymbol{\mu}_x^k\right)} = \\
&= \{\text{using variance properties}\} = \frac{2k\left(\mu_{x^Tx}^k - 2[\boldsymbol{\mu}_x^k]^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{x}\right)}{2k^2[\sigma_x^k]^2} = \\
&= \frac{[\sigma_x^k]^2 + [\boldsymbol{\mu}_x^k]^T\boldsymbol{\mu}_x - 2[\boldsymbol{\mu}_x^k]^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{x}}{k[\sigma_x^k]^2} = \frac{[\sigma_x^k]^2 + (\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k[\sigma_x^k]^2} = \\
&= \frac{[\sigma_x^k]^2 + (\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k[\sigma_x^k]^2} = \frac{1}{k} + \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k[\sigma_x^k]^2}.
\end{aligned} \tag{229}$$

The final eccentricity equation is

$$\xi^k(\boldsymbol{x}) = \frac{1}{k} + \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k[\sigma_x^k]^2}. \tag{230}$$

One can notice that the quantities $\boldsymbol{\mu}_x^k$ and σ_x^k can be calculated recursively using the following formulae [19]:

$$\boldsymbol{\mu}_x^k = \frac{k-1}{k}\boldsymbol{\mu}_x^{k-1} + \frac{\boldsymbol{x}_k}{k}, \boldsymbol{\mu}_x^0 = \boldsymbol{0}. \tag{231}$$

$$\mu_{x^Tx}^k = \frac{(k-1)\mu_{x^Tx}^{k-1}}{k} + \frac{\boldsymbol{x}_k^T\boldsymbol{x}_k}{k}, k \geq 1, \mu_{x^Tx}^0 = 0, \tag{232}$$

$$[\sigma_x^k]^2 = \mu_{x^Tx}^k - [\boldsymbol{\mu}_x^k]^T\boldsymbol{\mu}_x. \tag{233}$$

One can see from these equations that it is possible to update the mean and variance expressions and then recalculate eccentricity using the update values of the mean and variance.

The typicality can be represented using the formulae (224) and (230) as [213]

$$\tau^k(x) = 1 - \xi^k(x) = \frac{k-1}{k} - \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k[\sigma_x^k]^2}. \tag{234}$$

It gives an interpretation of the TEDA quantities in the case of Euclidean distances that relates it to the well-known Chebyshev inequality [160], [161]. Actually,

$$\begin{gathered}
t^k(x) = \frac{1}{k-2} - \frac{1}{k(k-2)} - \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k(k-2)[\sigma_x^k]^2} > \frac{1}{k}, \\
\frac{k-1}{k(k-2)} - \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k(k-2)[\sigma_x^k]^2} > \frac{1}{k}, \\
\frac{1}{k} + \frac{1}{k(k-2)} - \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k(k-2)[\sigma_x^k]^2} > \frac{1}{k}, \\
\frac{1}{k(k-2)} > \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k(k-2)[\sigma_x^k]^2}, \\
(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x}) < [\sigma_x^k]^2.
\end{gathered} \tag{235}$$

Expanding for "$m\sigma$" rule gives the equation

$$t^k(\boldsymbol{x}) = \frac{1}{k-2} - \frac{1}{k(k-2)} - \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{k(k-2)[\sigma_x^k]^2} > \frac{-m^2 + k - 1}{k(k-2)} = T(k), \tag{236}$$

For the normalised eccentricity similar equations can be obtained [19]:

$$(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x}) < m^2[\sigma_x^k]^2,$$

$$\frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^T(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{2k[\sigma_x^k]^2} < \frac{m^2}{2k}, \tag{237}$$

$$\zeta^k(\boldsymbol{x}) = \frac{1}{2k} + \frac{(\boldsymbol{\mu}_x^k - \boldsymbol{x})^{\boldsymbol{T}}(\boldsymbol{\mu}_x^k - \boldsymbol{x})}{2k[\sigma_x^k]^2} < \frac{m^2+1}{2k}.$$

4.1.2.2 *Mahalanobis case*

Similar results [19] as for the Euclidean distance can be obtained also for the case when Mahalanobis distance is used [162] but here we also need to update the covariance matrix. For some problems, Euclidean distance solution can result in a low accuracy as it does not take into account the disparity between different components. That is why more complex distances are to be considered. The Mahalanobis distance is an obvious option because it can be considered as a generalisation of the Euclidean distance incorporating covariance matrix [162]. In [19] the method was described, capable of incremental TEDA update with Mahalanobis distance.

$$\xi^k(\boldsymbol{x}) = 2\frac{\sum_{i=1}^k d(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{i=1}^k \sum_{j=1}^k d(\boldsymbol{x}_i, \boldsymbol{x}_j)} = 2\frac{\sum_{i=1}^k (\boldsymbol{x} - \boldsymbol{x}_i)^T\,[\Sigma_x^k]^{-1}(\boldsymbol{x} - \boldsymbol{x}_i)}{\sum_{i=1}^k \sum_{j=1}^k (\boldsymbol{x}_i - \boldsymbol{x}_j)^T\,[\Sigma_x^k]^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_j)}, \tag{238}$$

where Σ_x^k is a covariance matrix defined as

$$[\Sigma_x^k] = \frac{1}{k}\sum_{m=1}^{k}(\boldsymbol{x}_m - \boldsymbol{\mu}_x^k)(\boldsymbol{x}_m - \boldsymbol{\mu}_x^k)^T. \tag{239}$$

Then the typicality's denominator is expressed as

$$\sum_{i=1}^{k}\sum_{j=1}^{k} d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{\substack{i=1,\\ j=1}}^{k} (\boldsymbol{x}_i - \boldsymbol{x}_j)^T\,[\Sigma_x^k]^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_j). \tag{240}$$

Then

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{i=1}^{k}\sum_{j=1}^{k}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T\,[\Sigma_x^k]^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_j) = 2\sum_{i=1}^{k}\sum_{j=1}^{k}\boldsymbol{x}_i^T[\Sigma_x^k]^{-1}\boldsymbol{x}_i -$$
$$-2\sum_{i=1}^{k}\sum_{j=1}^{k}\boldsymbol{x}_i^T[\Sigma_x^k]^{-1}\boldsymbol{x}_j = \left[2k\sum_{i=1}^{k}\boldsymbol{x}_i^T[\Sigma_x^k]^{-1}\boldsymbol{x}_i\right] - 2k^2[\boldsymbol{\mu}_x^k]^T[\Sigma_x^k]^{-1}[\boldsymbol{\mu}_x^k] = \tag{241}$$

$$= \left[2k \sum_{i=1}^{k} \boldsymbol{x}_i^T \left[\frac{1}{k}\sum_{j=1}^{k} \boldsymbol{x}_j \boldsymbol{x}_j^T - [\boldsymbol{\mu}_x^k][\boldsymbol{\mu}_x^k]^T\right]^{-1} \boldsymbol{x}_i\right] -$$

$$-2k^2[\boldsymbol{\mu}_x^k]^T \left[\frac{1}{k}\sum_{j=1}^{k} \boldsymbol{x}_j \boldsymbol{x}_j^T - [\boldsymbol{\mu}_x^k][\boldsymbol{\mu}_x^k]^T\right]^{-1} [\boldsymbol{\mu}_x^k] =$$

$$= \sum \left[2k \left[\sum_{i=1}^{k} \boldsymbol{x}_i \boldsymbol{x}_i^T\right] \otimes \left[\frac{1}{k}\sum_{j=1}^{k} \boldsymbol{x}_j \boldsymbol{x}_j^T - [\boldsymbol{\mu}_x^k][\boldsymbol{\mu}_x^k]^T\right]^{-1}\right] -$$

$$-\sum \left[2k^2 \left[[\boldsymbol{\mu}_x^k][\boldsymbol{\mu}_x^k]^T\right] \otimes \left[\frac{1}{k}\sum_{j=1}^{k} \boldsymbol{x}_j \boldsymbol{x}_j^T - [\boldsymbol{\mu}_x^k][\boldsymbol{\mu}_x^k]^T\right]^{-1}\right] =$$

$$= \left\{ \mu_{\boldsymbol{x}\boldsymbol{x}^T} = \frac{1}{k}\sum_{j=1}^{k} \boldsymbol{x}_j \boldsymbol{x}_j^T \right\} =$$

$$= 2k^2 \sum \left(\left[\mu_{\boldsymbol{x}\boldsymbol{x}^T} - \boldsymbol{\mu}_x^k [\boldsymbol{\mu}_x^k]^T\right] \otimes \left[\mu_{\boldsymbol{x}\boldsymbol{x}^T} - \boldsymbol{\mu}_x^k [\boldsymbol{\mu}_x^k]^T\right]^{-1} \right) =$$

$$= \left\{ \begin{matrix} \text{using} \\ \text{covariance matrix} \\ \text{symmetricity} \end{matrix} \right\} =$$

$$= 2k^2 \sum \left[\mu_{\boldsymbol{x}\boldsymbol{x}^T} - \boldsymbol{\mu}_x^k [\boldsymbol{\mu}_x^k]^T\right]\left[\mu_{\boldsymbol{x}\boldsymbol{x}^T} - \boldsymbol{\mu}_x^k [\boldsymbol{\mu}_x^k]^T\right]^{-1} \otimes I_{n\times n} =$$

$$= 2k^2 \operatorname{tr}(I_{n\times n}) = 2k^2 n,$$

where $\otimes$ denotes an element-wise matrix multililication operator, and $\sum[\,]$ is a sum over all matrix elements.

It shows that the denominator is dependent only of the sample set size k that significantly simplifies the problem. It leads to the interpretation of the eccentricity as a quantity proportional to the sum of the Mahalanobis distances. The equation for typicality is then written as

$$\xi^k(\boldsymbol{x}) = \frac{\sum_{i=1}^{k} (\boldsymbol{x} - \boldsymbol{x}_i)^T [\Sigma_x^k]^{-1} (\boldsymbol{x} - \boldsymbol{x}_i)}{k^2 n} =$$
$$= \frac{\boldsymbol{x}^T [\Sigma_x^k]^{-1} \boldsymbol{x} - 2[\boldsymbol{\mu}_x^k]^T [\Sigma_x^k]^{-1} \boldsymbol{x} + [\boldsymbol{\mu}_x^k]^T [\Sigma_x^k]^{-1} \boldsymbol{\mu}_x^k}{kn} + \qquad (242)$$

$$+\frac{\sum_{i=1}^{k}[\boldsymbol{x}_i]^T[\Sigma_x^k]^{-1}\boldsymbol{x}_i - k[\boldsymbol{\mu}_x^k]^T\,[\Sigma_x^k]^{-1}\boldsymbol{\mu}_x^k}{k^2 n} =$$

$$= \frac{\boldsymbol{x}^T\,[\Sigma_x^k]^{-1}\boldsymbol{x} - 2[\boldsymbol{\mu}_x^k]^T[\Sigma_x^k]^{-1}\boldsymbol{x} + [\boldsymbol{\mu}_x^k]^T[\Sigma_x^k]^{-1}\boldsymbol{\mu}_x^k}{kn} + \frac{1}{k} =$$

$$= \frac{(\boldsymbol{x} - \boldsymbol{\mu}_x^k)^T\,[\Sigma_x^k]^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x^k)}{kn} + \frac{1}{k}.$$

Then the conditions, imposed on the typicality, corresponding to Chebyshev inequality can be described as:

$$(\boldsymbol{x} - \boldsymbol{\mu}_x^k)^T[\Sigma_x^k]^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x^k) < m^2, \tag{243}$$

where m is a threshold. The normalised eccentricity can be expressed as

$$\zeta^k(\boldsymbol{x}) = \frac{(\boldsymbol{x} - \boldsymbol{\mu}_x^k)^T[\Sigma_x^k]^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x^k)}{2kn} + \frac{1}{2k} < \frac{m^2 + n}{2kn}, \tag{244}$$

$$\zeta^k(\boldsymbol{x}) < \frac{m^2 + n}{2kn}. \tag{245}$$

The analogous equations can be given for typicality [19]:

$$\begin{gathered} t^k(\boldsymbol{x}) = \frac{1}{k-1} - \frac{(\boldsymbol{x} - \boldsymbol{\mu}_x^k)^T[\Sigma_x^k]^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x^k)}{kn(k-1)} - \frac{1}{kn(k-1)} > \\ > \frac{1}{k-1} - \frac{m^2 + n}{kn(k-1)} = T(k). \end{gathered} \tag{246}$$

4.1.3 Covariance matrix update

In this section, the recursive update formulae for the covariance matrices are given. The distance on the k-th step is defined as a scalar by the formula

$$d^k(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu}_x)^T[\Sigma_x^k]^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_x). \tag{247}$$

The covariance matrix is expressed as

$$\begin{gathered} \Sigma_x^k = \frac{1}{k}\sum_{i=1}^{k}(\boldsymbol{x}_i - \boldsymbol{\mu}_x^k)(\boldsymbol{x}_i - \boldsymbol{\mu}_x^k)^T = \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{x}_i\boldsymbol{x}_i^T - \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{\mu}_x^k[\boldsymbol{\mu}_x^k]^T = \\ = \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{x}_i\boldsymbol{x}_i^T - \boldsymbol{\mu}_x^k{\boldsymbol{\mu}_x^k}^T. \end{gathered} \tag{248}$$

Further in this section, the first term of the covariance matrix expression is denoted as

$$\mu_{xx^T}^{k} = \frac{1}{k}\sum_{j=1}^{k} \boldsymbol{x}_j \boldsymbol{x}_j^T. \tag{249}$$

The mean value is updated from the step $(k-1)$ to the step k:

$$\boldsymbol{\mu}_x^k = \frac{k-1}{k}\boldsymbol{\mu}_x^{k-1} + \frac{\boldsymbol{x}_k}{k}, \boldsymbol{\mu}_x^0 = \mathbf{0}. \tag{250}$$

The similar update procedure can be provided for $\mu_{xx^T}^k$:

$$\mu_{xx^T}^{k} = \frac{k-1}{k}\mu_{xx^T}^{k-1} + \frac{1}{k}\boldsymbol{x}_k \boldsymbol{x}_k^T, \tag{251}$$

Then the final expression for the covariance matrix update is

$$\begin{aligned}\Sigma_x^k &= \frac{k-1}{k}\boldsymbol{\mu}_{xx^T}^{k-1} + \frac{\boldsymbol{x}_k\boldsymbol{x}_k^T}{k} - \left(\frac{k-1}{k}\boldsymbol{\mu}_x^{k-1} + \frac{\boldsymbol{x}_k}{k}\right)\left(\frac{k-1}{k}\boldsymbol{\mu}_x^{k-1} + \frac{\boldsymbol{x}_k}{k}\right)^T = \\ &= \frac{k-1}{k}\Sigma_x^{k-1} + \frac{k^2-1}{k^2}\,\boldsymbol{x}_k\boldsymbol{x}_k^T + \frac{k-1}{k^2}\boldsymbol{\mu}_x^{k-1}([\boldsymbol{\mu}_x^{k-1}] - 2\boldsymbol{x}_k)^T\,.\end{aligned} \tag{252}$$

To make such update computationally efficient, matrix inversions are to be avoided for every new data sample. Fortunately, there is a way to perform the inverse covariance matrix update. It is based on well-known Woodbury formula [163]:

$$(A + UBV)^{-1} = A^{-1} - A^{-1}U(B^{-1} + VA^{-1}U)^{-1}VA^{-1}, \tag{253}$$

where A, B, U, V are the matrices with sizes compatible for matrix multiplication and sum, and A is an invertible square matrix. To perform the inference, this formula should be applied twice: at the first stage, for the first and the second terms of the recursive expression (252), and, after that, for the sum of the first and the second terms, and the third term.

First, p is introduced as

$$p = \frac{k-1}{k}\Sigma_{\mathrm{x}}^{k-1} + \frac{k^2-1}{k^2}\,\boldsymbol{x}_k\boldsymbol{x}_k^T. \tag{254}$$

Then the following matrices are substituted into the Woodbury formula [163]:

$$A^{-1} = \frac{k}{k-1}[\Sigma_x^{k-1}]^{-1}, B = 1, U = \frac{k^2-1}{k^2}\boldsymbol{x}_k, V = \boldsymbol{x}_{k+1}^T. \tag{255}$$

Then p^{-1} can be expressed as

$$p^{-1} = \frac{k[\Sigma_x^{k-1}]^{-1}}{k-1} + \frac{k[\Sigma_x^{k-1}]^{-1}}{k-1}\frac{k+1}{k}\boldsymbol{x}_k\left(1 + \frac{k+1}{k}\boldsymbol{x}_k^T[\Sigma_x^{k-1}]^{-1}\boldsymbol{x}_k\right)^{-1} \times \tag{256}$$

$$\times \boldsymbol{x}_k^T[\Sigma_x^{k-1}]^{-1} = \frac{k}{k-1}[\Sigma_x^{k-1}]^{-1} + \frac{\frac{k}{k-1}[\Sigma_x^{k-1}]^{-1}\frac{k+1}{k}\boldsymbol{x}_k\boldsymbol{x}_k^T[\Sigma_x^{k-1}]^{-1}}{1+\frac{k+1}{k}\boldsymbol{x}_k^T[\Sigma_x^{k-1}]^{-1}\boldsymbol{x}_k},$$

$$\Sigma_x^k = p + \frac{k-1}{k^2}\boldsymbol{\mu}_x^{k-1}(\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k)^T .$$

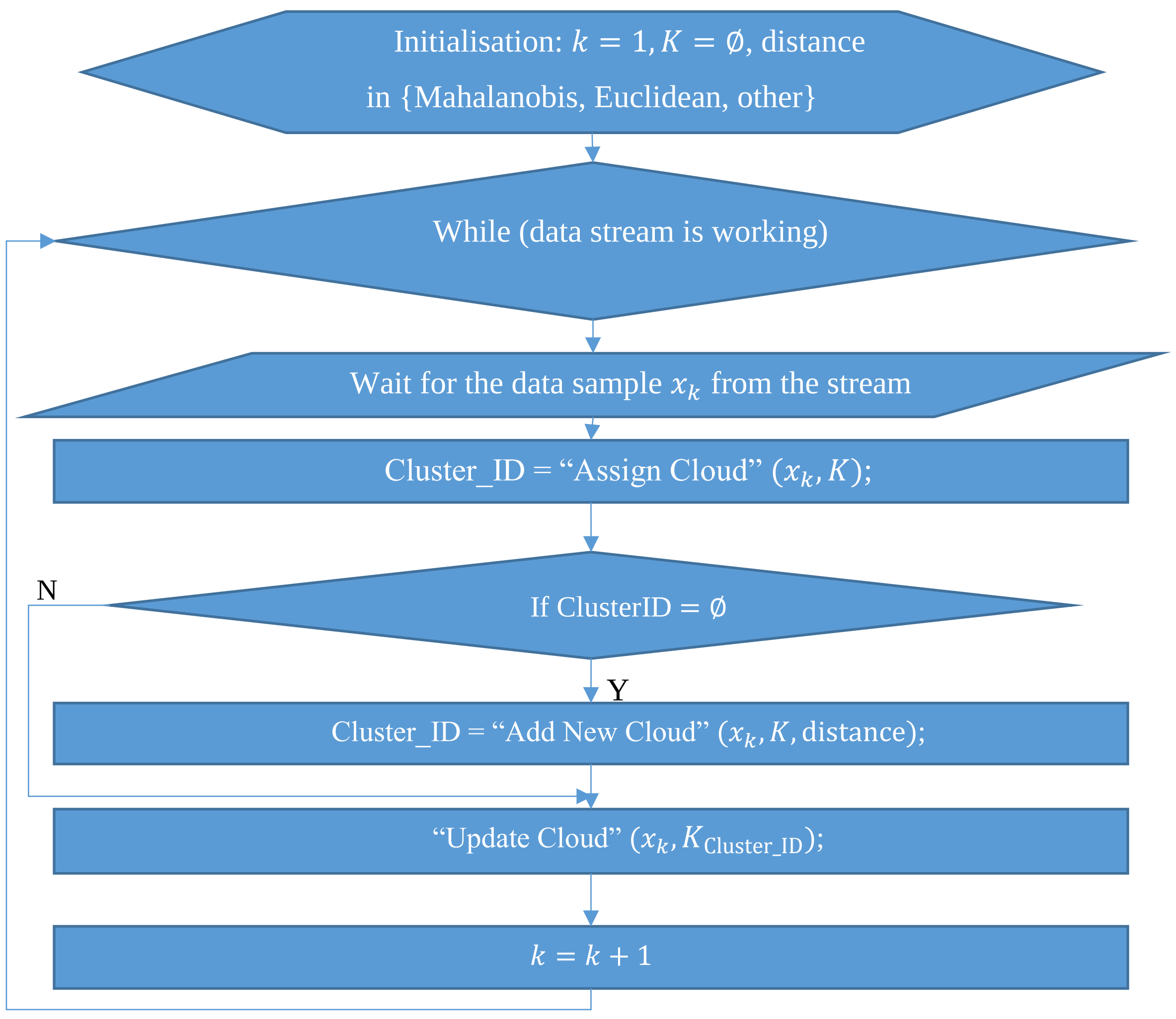
After the second substitution is made, so that $A^{-1} = p^{-1}, B = 1, U = \frac{k-1}{k^2}\boldsymbol{\mu}_x^{k-1}, V = (\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k)^T$. The incremental update of the covariance matrix is then made as follows:

$$[\Sigma_x^k]^{-1} = p^{-1} + p^{-1}\frac{(k-1)\boldsymbol{\mu}_x^k}{k^2}\left(1+\frac{k-1}{k^2}(\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k)^T p^{-1}\boldsymbol{\mu}_x^k\right)^{-1} \times$$
$$\times (\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k)^T p^{-1} = p^{-1} + \frac{p^{-1}\frac{(k-1)\boldsymbol{\mu}_x^{k-1}}{k^2}(\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k)^T p^{-1}}{1+\frac{(k-1)}{k^2}\left(\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k\right)^T p^{-1}\boldsymbol{\mu}_x^{k-1}}. \quad (257)$$

The final exact incremental update formula is then [163]

$$[\Sigma_x^k]^{-1} = p^{-1} + \frac{p^{-1}\frac{(k-1)\boldsymbol{\mu}_x^{k-1}}{k^2}\left(\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k\right)^T p^{-1}}{1+\frac{(k-1)}{k^2}\left(\boldsymbol{\mu}_x^{k-1} - 2\boldsymbol{x}_k\right)^T p^{-1}\boldsymbol{\mu}_x^{k-1}}. \quad (258)$$

4.1.4 TEDACluster



Initialisation: $k = 1, K = \emptyset$, distance in {Mahalanobis, Euclidean, other}
While (data stream is working)
Wait for the data sample x_k from the stream
Cluster_ID = "Assign Cloud" (x_k, K);
N
If ClusterID $= \emptyset$
Y
Cluster_ID = "Add New Cloud" $(x_k, K,$ distance$)$;
"Update Cloud" $(x_k, K_{\text{Cluster\_ID}})$;
$k = k + 1$

Figure 13 Main TEDACluster processing flow

Based on the incremental update procedures, described in the previous sections, the novel clustering approach is proposed, based on TEDA and referred to as TEDACluster [19]. The outline of the algorithm is depicted in Figure 13. The approach is based on the AnYa fuzzy rule system [111], which deals with data clouds. Shapes and boundaries of data clouds are not pre-defined but built based on data pattern alone.

The AnYa fuzzy rule system F, composed of the fuzzy rules $R_i, i = \overline{1 \dots N}$, is defined for clustering as [111]

$$F = \{R_i\}, i = \overline{1, N},$$
$$R_{\text{i}}(\boldsymbol{x}): IF\ (\boldsymbol{x} \sim \boldsymbol{x}_i^*)\ THEN\ i. \tag{259}$$

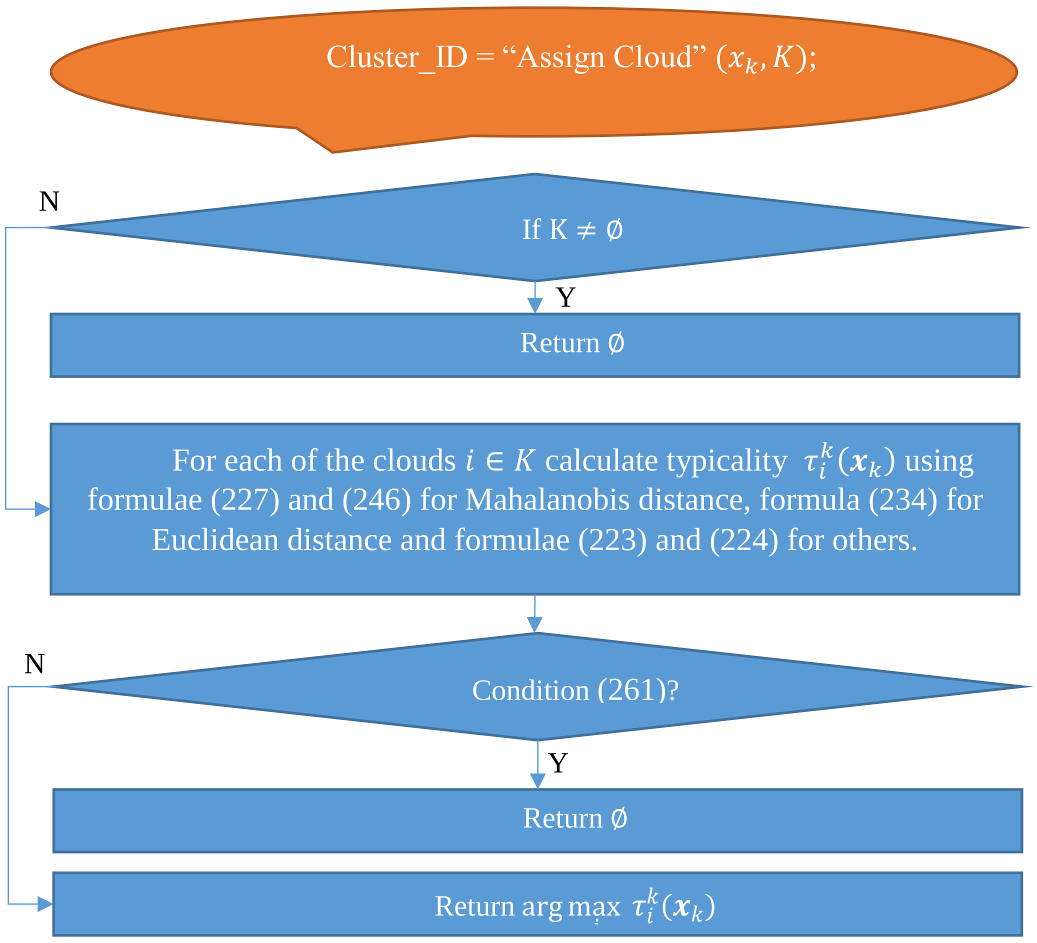

Cluster_ID = "Assign Cloud" (x_k, K);
N
If K ≠ ∅
Y
Return ∅
For each of the clouds $i \in K$ calculate typicality $\tau_i^k(\boldsymbol{x}_k)$ using formulae (227) and (246) for Mahalanobis distance, formula (234) for Euclidean distance and formulae (223) and (224) for others.
N
Condition (261)?
Y
Return ∅
Return $\arg\max_i \tau_i^k(\boldsymbol{x}_k)$

Figure 14 Procedure "Assign Cloud" of TEDACluster algorithm

Each fuzzy rule is interpreted, henceforth, as a cluster. Here, $\boldsymbol{x}, \boldsymbol{x}_i^* \in \mathfrak{X}, F: \mathfrak{X} \rightarrow K,\ \mathfrak{X}$ is a feature space, K is the finite set of clusters, $\boldsymbol{x}_i^*$ is the cluster's focal point, $\sim$ is a like predicate, i.e. some closeness relation over the feature space. The new vectors are assigned to the clusters using their typicality with respect to each of the clouds (see Figure 14). We should mention here that the "like" predicate, "$\sim$", still needs to be defined. Then, the way fuzzy rules are added or deleted, and the algorithms should be defined. The predicate "$\sim$" is defined as a firing strength for each rule, so that

$$w_i^k(\boldsymbol{x}) = \frac{t_i^k(\boldsymbol{x})}{\sum_{j=1}^{N} t_j^k(\boldsymbol{x})}, \tag{260}$$

where $t_i^k(\boldsymbol{x})$ is a normalised typicality of $\boldsymbol{x}$ over all the data. After that, the rule addition and deletion procedures are to be defined for the fuzzy rule system. In this clustering technique no global statistical data characteristics are calculated, but each of the clusters have their own descriptors instead. It differs from the approach applied in eClass [114] and AutoClass [17].

A new data cloud formation condition is given as follows:

$$\forall\, R^i \in F\ t_i^k(\boldsymbol{x}) < T(k) \Leftrightarrow \mathfrak{Y}(k) = 1, \tag{261}$$

where $\mathfrak{Y}(k)$ is a flag, showing that the new cloud should be created, $T(k)$ is a threshold depending of the data granularity given by the equations (236) and (246) for the Euclidean and Mahalanobis distances correspondingly. The algorithm for the cluster addition is formulated in Figure 15.

After the addition of each new data cloud, all clouds are assessed in regards to whether they are close to each other. It means that if for any of the clouds exists such a cluster which typicality is greater than $\kappa T(k), \kappa > 0,$ then it is assumed that the cluster needs to be merged into this cloud:

$$R^i\colon \exists\, R^j \in F\colon i \neq j, t_j^k(\boldsymbol{\mu}^i) < \kappa T(k) \Leftrightarrow \mathrm{merge}\left(R^i, R^j\right), \tag{262}$$

where $\mathrm{merge}\left(R^i, R^j\right)$ is a data cloud merging operation. Removal does not mean that the cloud is deleted completely, but means instead that two or more clouds are merged and, as a result, one get a united cloud. During the merging stage, the mean is updated as

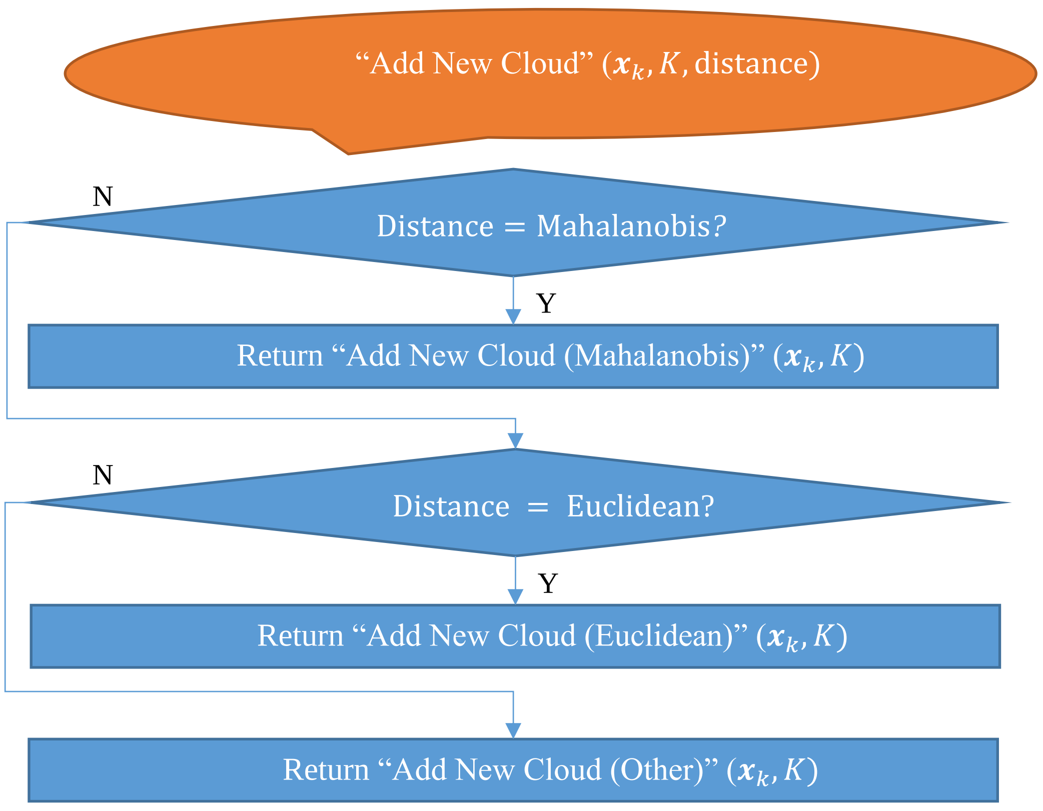
$$\boldsymbol{\mu}_l^k = \frac{\mathrm{Support}(R_\mathrm{i})\boldsymbol{\mu}_i^k}{\mathrm{Support}(R_\mathrm{i}) + \mathrm{Support}(R_\mathrm{j})} + \frac{\mathrm{Support}(R_\mathrm{j})\boldsymbol{\mu}_j^k}{\mathrm{Support}(R_\mathrm{i}) + \mathrm{Support}(R_\mathrm{j})}, k \geq 1. \tag{263}$$

The function $\mathrm{Support}(R_\mathrm{i})$ denotes the cloud support, i.e. a number of the objects assigned to the clusters to contribute to its mean (see Figure 16). To update the variance, we need also another formula to update the mean squared scalar product:

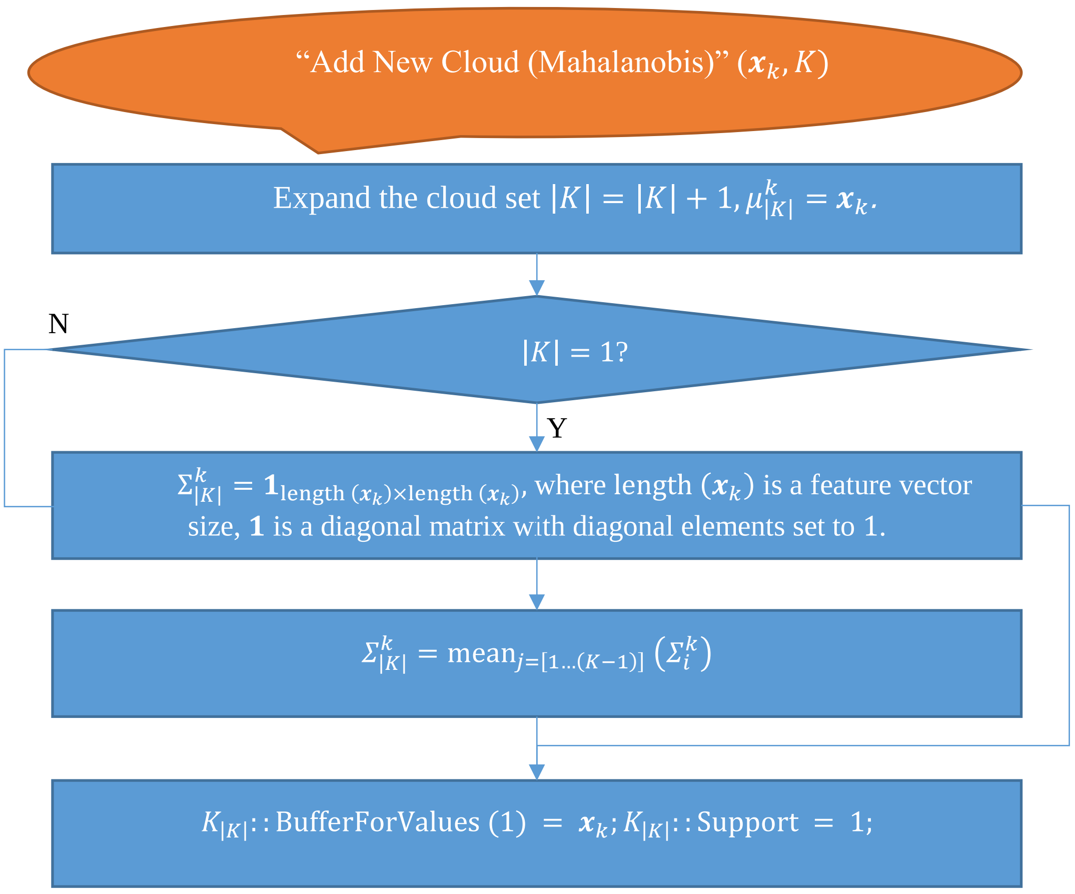
$$\boldsymbol{\mu}_{x^Tx,l}^k = \frac{\mathrm{Support}(R_\mathrm{i})\boldsymbol{\mu}_{x^Tx,i}^k}{\mathrm{Support}(R_\mathrm{i}) + \mathrm{Support}(R_\mathrm{j})} + \frac{\mathrm{Support}(R_\mathrm{j})\boldsymbol{\mu}_{x^Tx,j}^k}{\mathrm{Support}(R_\mathrm{i}) + \mathrm{Support}(R_\mathrm{j})}, k \geq 1. \tag{264}$$

The inverted covariance, in case of Mahalanobis distance, can be updated via the Woodbury's formula [163].

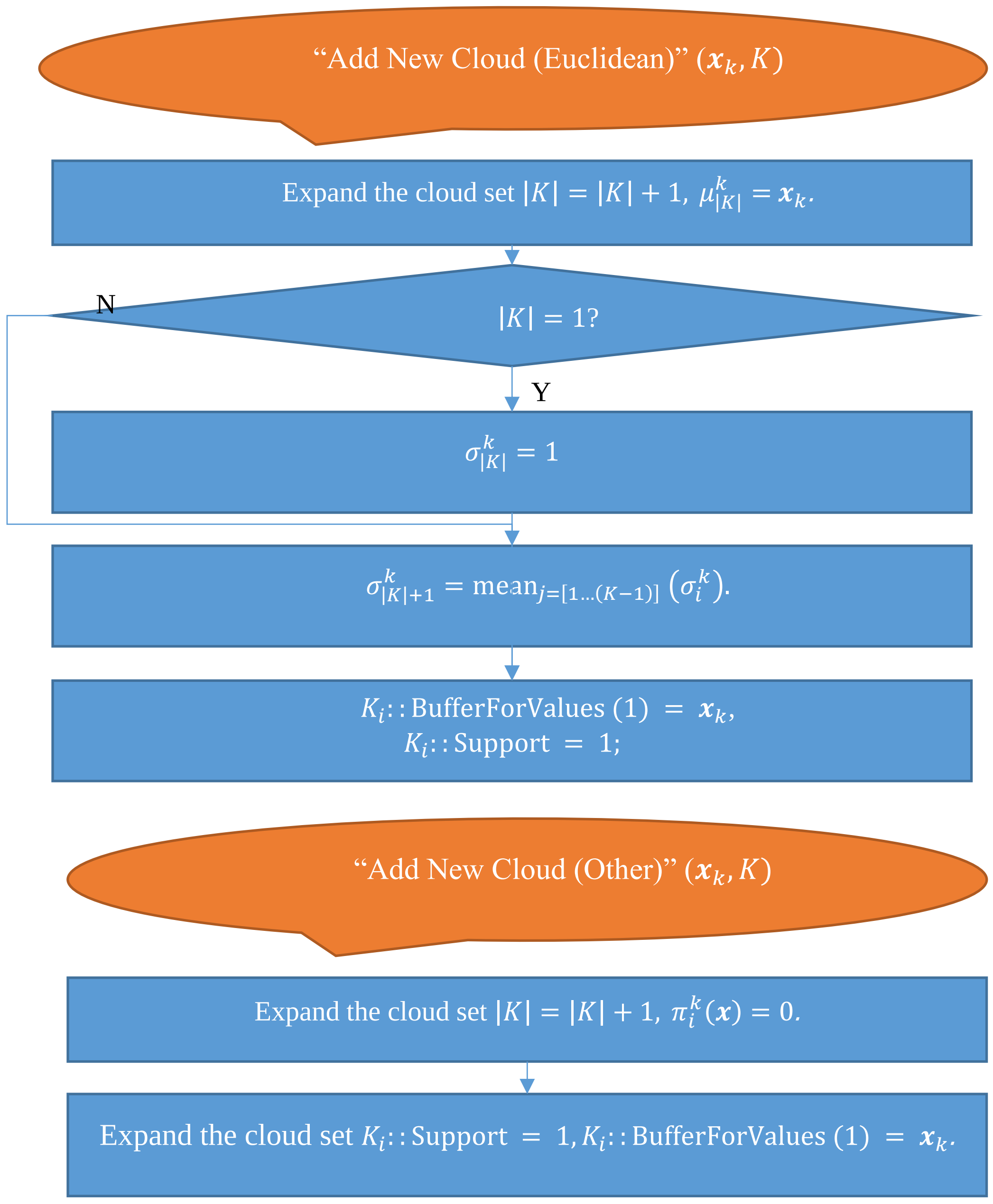
The experimental results for this method is given in section 5.2.



“Add New Cloud” ($\boldsymbol{x}_k, K$, distance)
Distance = Mahalanobis?
N
Y
Return “Add New Cloud (Mahalanobis)” ($\boldsymbol{x}_k, K$)
Distance = Euclidean?
N
Y
Return “Add New Cloud (Euclidean)” ($\boldsymbol{x}_k, K$)
Return “Add New Cloud (Other)” ($\boldsymbol{x}_k, K$)

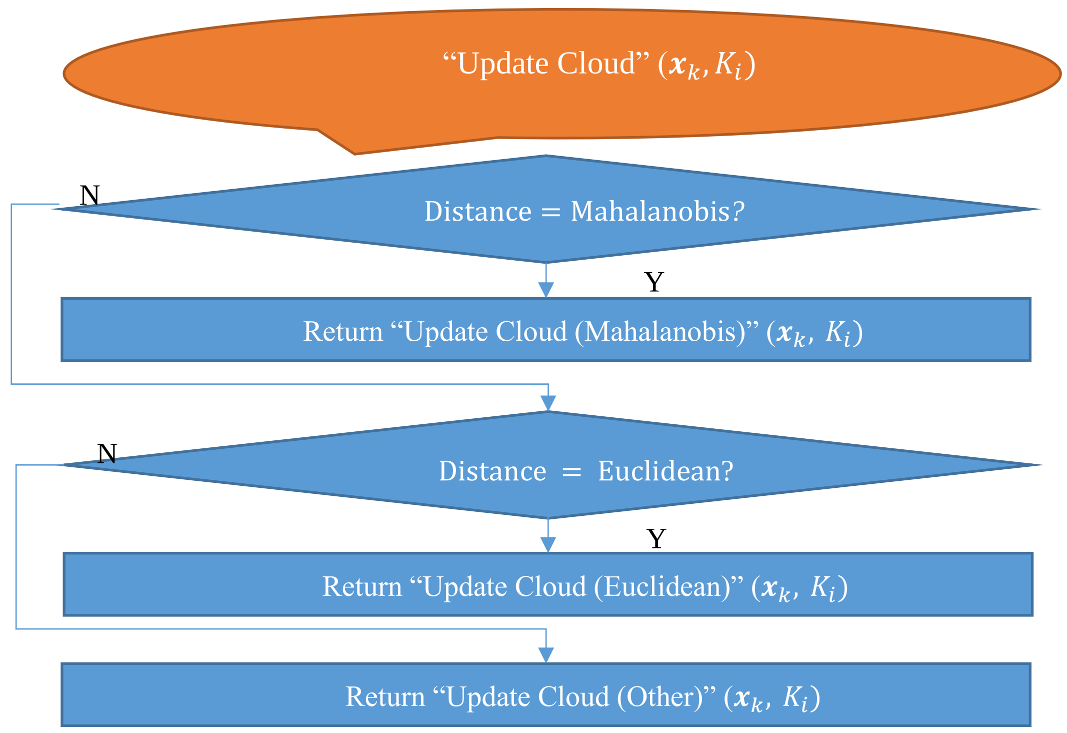


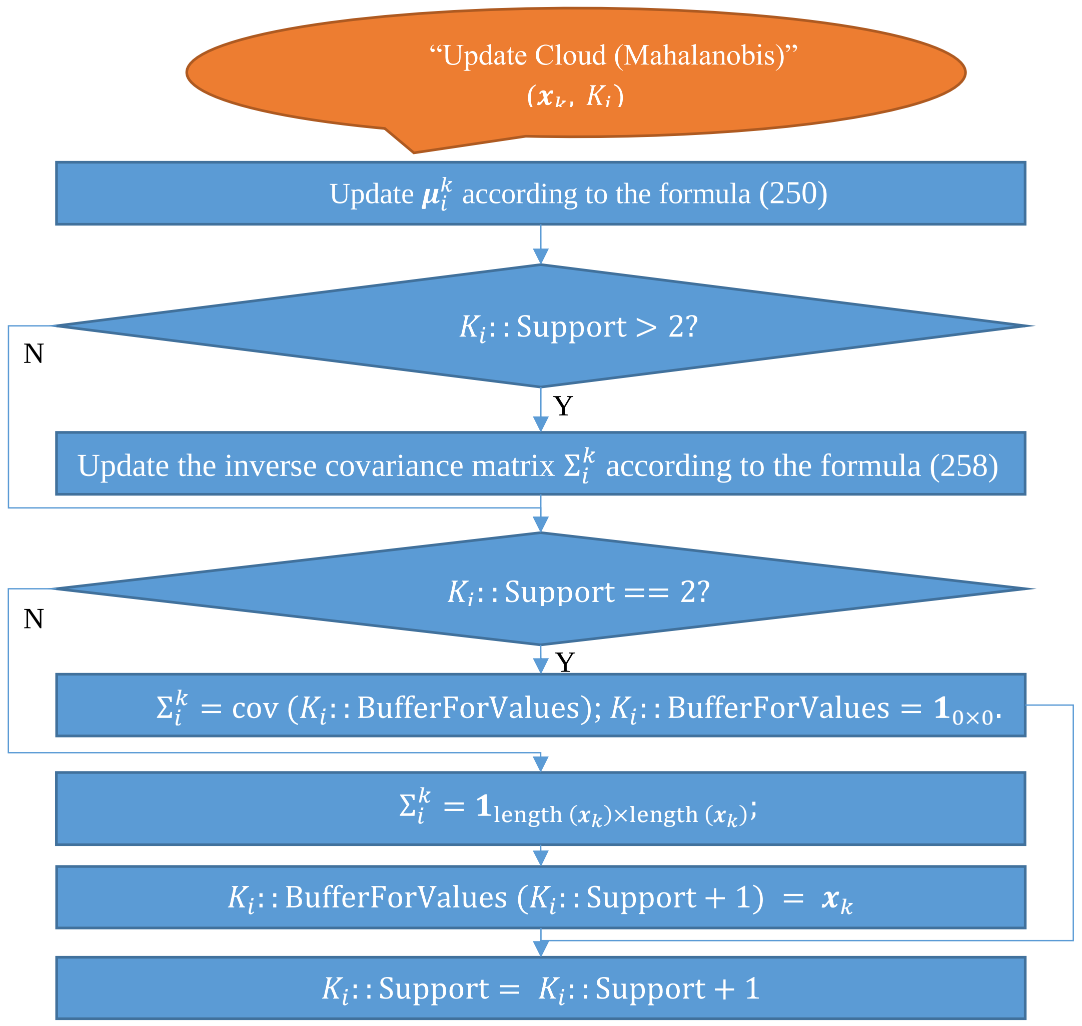
“Add New Cloud (Mahalanobis)” ($\boldsymbol{x}_k, K$)
Expand the cloud set $|K| = |K| + 1, \mu_{|K|}^k = \boldsymbol{x}_k$.
$|K| = 1$?
N
Y
$\Sigma_{|K|}^k = \mathbf{1}_{\text{length}\,(x_k)\times\text{length}\,(x_k)}$, where length $(\boldsymbol{x}_k)$ is a feature vector size, $\mathbf{1}$ is a diagonal matrix with diagonal elements set to 1.
$\Sigma_{|K|}^k = \text{mean}_{j=[1\ldots(K-1)]}\left(\Sigma_i^k\right)$
$K_{|K|}$::BufferForValues (1) = $\boldsymbol{x}_k$; $K_{|K|}$::Support = 1;



“Add New Cloud (Euclidean)” $(\boldsymbol{x}_k, K)$
Expand the cloud set $|K| = |K| + 1,\ \mu_{|K|}^{k} = \boldsymbol{x}_k.$
$|K| = 1?$
N
Y
$\sigma_{|K|}^{k} = 1$
$\sigma_{|K|+1}^{k} = \text{mean}_{j=[1\ldots(K-1)]}\left(\sigma_i^k\right).$
$K_i{::}\text{BufferForValues}\ (1) = \boldsymbol{x}_k,$
$K_i{::}\text{Support} = 1;$
“Add New Cloud (Other)” $(\boldsymbol{x}_k, K)$
Expand the cloud set $|K| = |K| + 1,\ \pi_i^k(\boldsymbol{x}) = 0.$
Expand the cloud set $K_i{::}\text{Support} = 1, K_i{::}\text{BufferForValues}\ (1) = \boldsymbol{x}_k.$

Figure 15 Procedure “Add New Cloud” of TEDACluster algorithm


“Update Cloud” $(\boldsymbol{x}_k, K_i)$
Distance = Mahalanobis?
N
Y
Return “Update Cloud (Mahalanobis)” $(\boldsymbol{x}_k,\ K_i)$
Distance = Euclidean?
N
Y
Return “Update Cloud (Euclidean)” $(\boldsymbol{x}_k,\ K_i)$
Return “Update Cloud (Other)” $(\boldsymbol{x}_k,\ K_i)$


“Update Cloud (Mahalanobis)” $(\boldsymbol{x}_k,\ K_i)$
Update $\boldsymbol{\mu}_i^k$ according to the formula (250)
$K_i::\text{Support} > 2?$
N
Y
Update the inverse covariance matrix Σ_i^k according to the formula (258)
$K_i::\text{Support} == 2?$
N
Y
$\Sigma_i^k = \text{cov}\ (K_i::\text{BufferForValues});\ K_i::\text{BufferForValues} = \mathbf{1}_{0\times 0}.$
$\Sigma_i^k = \mathbf{1}_{\text{length}\ (\boldsymbol{x}_k)\times\text{length}\ (\boldsymbol{x}_k)};$
$K_i::\text{BufferForValues}\ (K_i::\text{Support} + 1) = \boldsymbol{x}_k$
$K_i::\text{Support} = K_i::\text{Support} + 1$

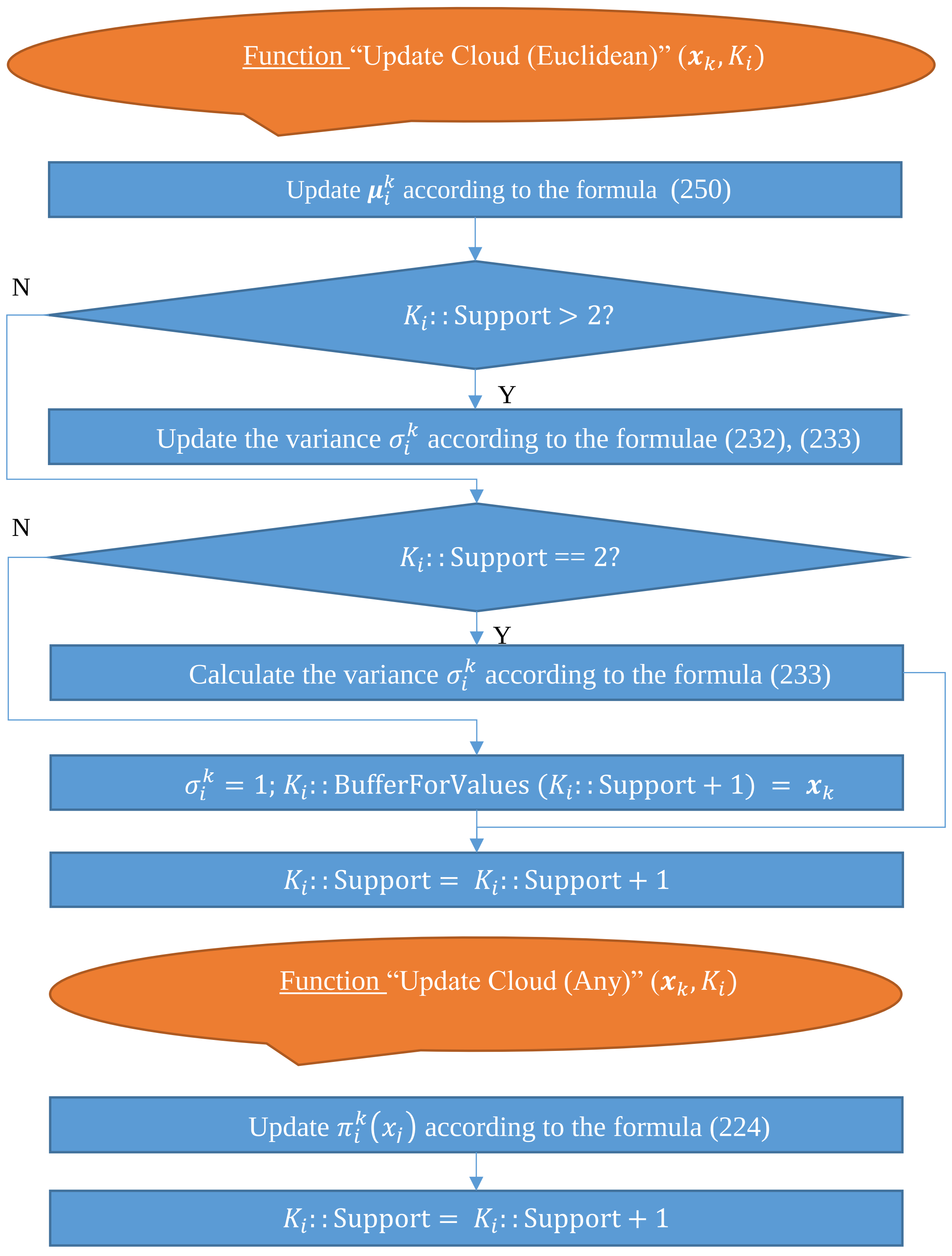

Function "Update Cloud (Euclidean)" $(\boldsymbol{x}_k, K_i)$
Update $\boldsymbol{\mu}_i^k$ according to the formula (250)
N
$K_i::\text{Support} > 2?$
Y
Update the variance σ_i^k according to the formulae (232), (233)
N
$K_i::\text{Support} == 2?$
Y
Calculate the variance σ_i^k according to the formula (233)
$\sigma_i^k = 1; K_i::\text{BufferForValues}\ (K_i::\text{Support} + 1) = \boldsymbol{x}_k$
$K_i::\text{Support} = K_i::\text{Support} + 1$
Function "Update Cloud (Any)" $(\boldsymbol{x}_k, K_i)$
Update $\pi_i^k(x_j)$ according to the formula (224)
$K_i::\text{Support} = K_i::\text{Support} + 1$

Figure 16 Procedure "Update Cloud"

4.2 Classification and regression techniques

4.2.1 TEDAClass

Based on the TEDACluster algorithm, described in section 4.1, the TEDA classifier is proposed with attention to the important particular cases of Euclidean and Mahalanobis distances. As it was explained before, in section 4.1, these distances have attractive mathematical properties of the solution which enables an online update procedure.

As it was discussed in section 2.2.5, AnYa fuzzy rule system was introduced first in [111]. Let F be a fuzzy rule set, composed of N rules R_i. The rule output set is denoted C, rule outputs $y_i \in C$. Each rule is described by some representative point $\boldsymbol{x}_i^*$. Using this notation AnYa fuzzy rule set is defined as

$$F = \{R_i\}, i = \overline{1, N}, \tag{265}$$

$$R_{\rm i}(\boldsymbol{x}): IF\ (\boldsymbol{x} \sim \boldsymbol{x}_i^*)\ THEN\ y_i = \overline{\boldsymbol{x}}^T \Theta_i. \tag{266}$$

where $\sim$ is some similarity relation for the input to the representative point.

For each sample vector, the fuzzy rule set is being changed in evolving way that means that we can merge existing or add new fuzzy rules. The training of TEDAClass is similar to the one of AutoClass [17] and eClass [115], but the fuzzy rules are based on TEDA instead. This impacts the definition of the similarity relation and new rules creation criteria. More precisely, we base the closeness to the rule on the typicality: the more typical is the point, the better is the vector $\boldsymbol{x}$ described by this rule. The rule's firing strength is defined given the normalised typicality $t_i^k\ (\boldsymbol{x})$ within the i-th rule as [19], [20]

$$w_i^k(\boldsymbol{x}) = \frac{t_i^k(\boldsymbol{x})}{\sum_{j=1}^{N} t_j^k(\boldsymbol{x})}. \tag{267}$$

The final result is given as a weighted sum of the results of each individual rule, weighed with the firing strengths.

The rule creation criterion is given as follows. The new rule is created when the local typicality does not exceed the threshold given by the equation given by the equations (236) and (246) for the Euclidean and Mahalanobis distances correspondingly [19], [20]:

$$\forall\, R_i \in F\ t_i^k(\boldsymbol{x}) < T(k) \Leftrightarrow \mathfrak{Y}(k) = 1, \tag{268}$$

where $\mathfrak{Y}(k) \in \{0,1\}$ is a flag, raised to 1 if the cluster should be created, and taking the zero value otherwise.

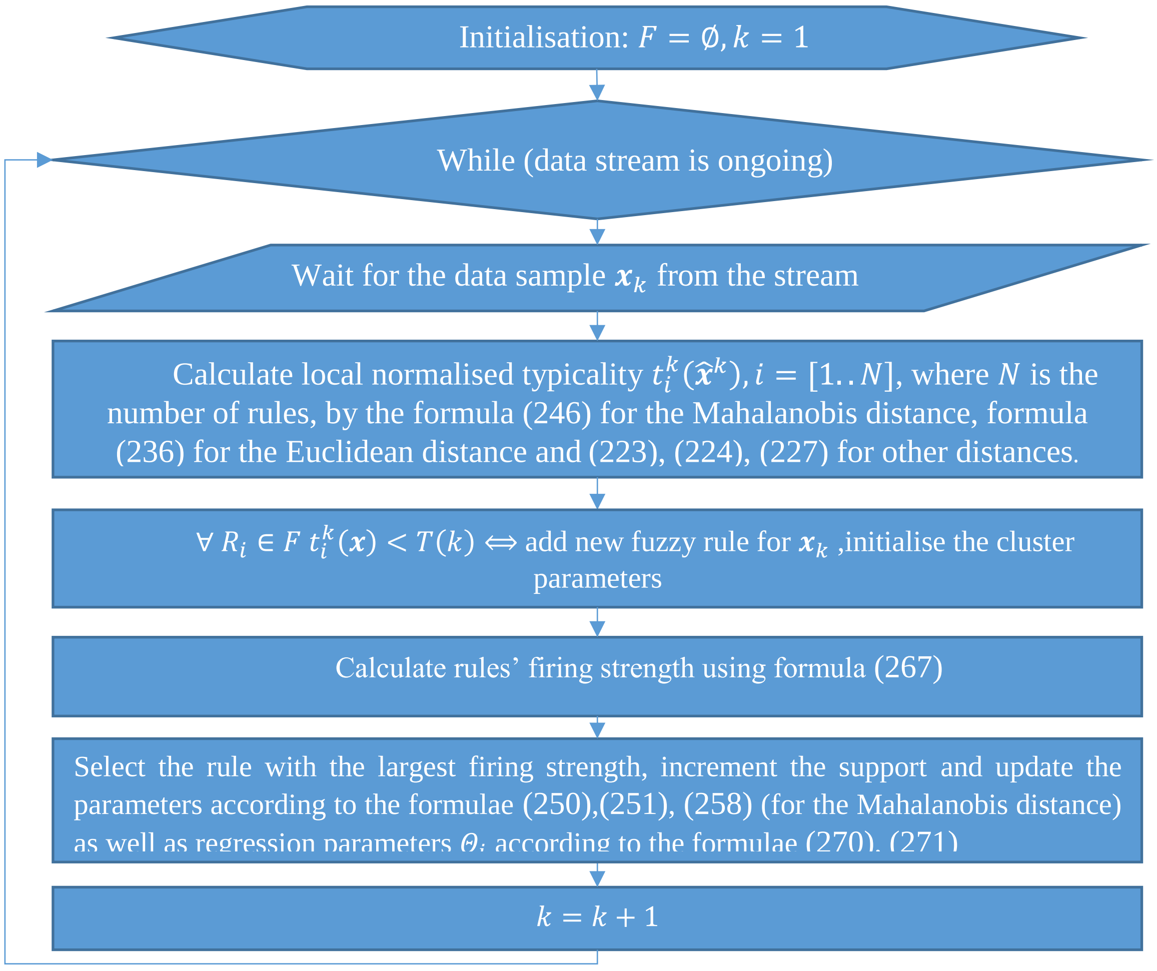

Initialisation: $F = \emptyset, k = 1$
While (data stream is ongoing)
Wait for the data sample $\boldsymbol{x}_k$ from the stream
Calculate local normalised typicality $t_i^k(\hat{\boldsymbol{x}}^k), i = [1..N]$, where N is the number of rules, by the formula (246) for the Mahalanobis distance, formula (236) for the Euclidean distance and (223), (224), (227) for other distances.
$\forall R_i \in F\ t_i^k(\boldsymbol{x}) < T(k) \Leftrightarrow$ add new fuzzy rule for $\boldsymbol{x}_k$,initialise the cluster parameters
Calculate rules' firing strength using formula (267)
Select the rule with the largest firing strength, increment the support and update the parameters according to the formulae (250),(251), (258) (for the Mahalanobis distance) as well as regression parameters Θ_i according to the formulae (270), (271)
$k = k + 1$

Figure 17 TEDAClass training algorithm

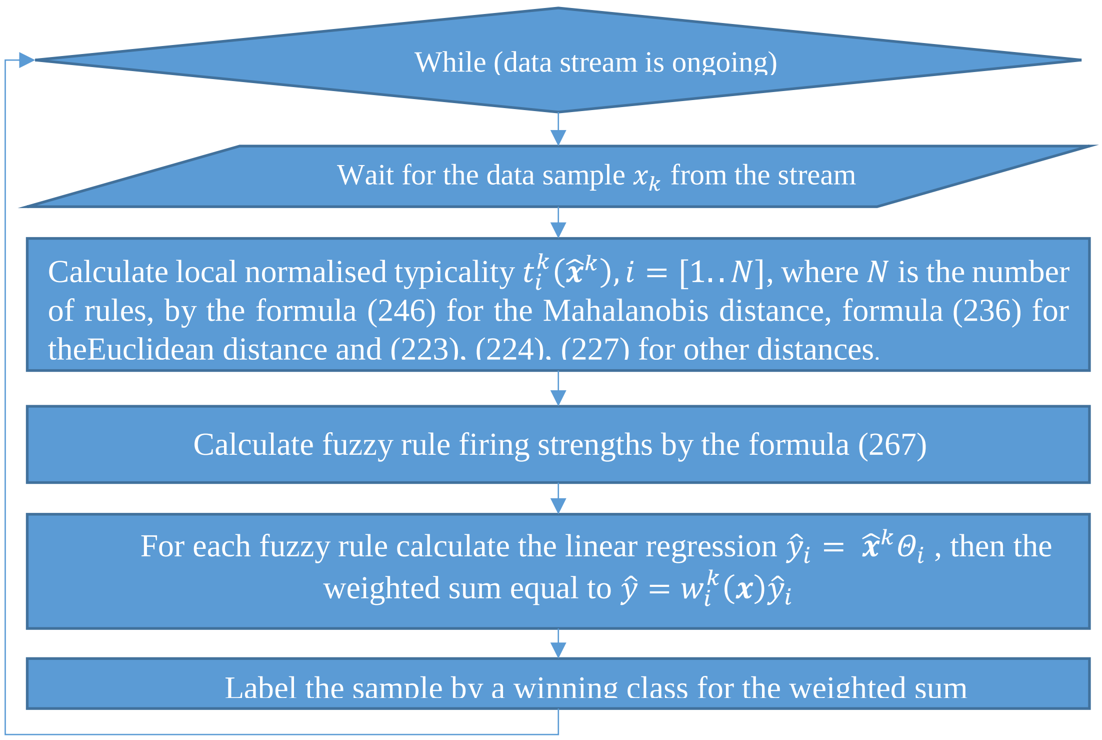

While (data stream is ongoing)
Wait for the data sample x_k from the stream
Calculate local normalised typicality $t_i^k(\hat{\boldsymbol{x}}^k), i = [1..N]$, where N is the number of rules, by the formula (246) for the Mahalanobis distance, formula (236) for theEuclidean distance and (223), (224), (227) for other distances.
Calculate fuzzy rule firing strengths by the formula (267)
For each fuzzy rule calculate the linear regression $\hat{y}_i = \hat{\boldsymbol{x}}^k \Theta_i$, then the weighted sum equal to $\hat{y} = w_i^k(\boldsymbol{x})\hat{y}_i$
Label the sample by a winning class for the weighted sum

Figure 18 TEDAClass recognition algorithm

One can note that the new rules are being created using only local quantities, unlike the approach adopted by the allied classifiers eClass [115] and AutoClass [17]. Local typicality is the typicality for the points-participants of a single data cloud. After the new rule is added, the similar rules closer than $T(k)$ are deleted. It means that the set of rules for deletion is [19], [20]

$$R_d = \left\{R_i \in F: t_i^k(\boldsymbol{x}_k) > T(k)\right\}. \tag{269}$$

The training algorithm is formalised into the scheme depicted in Figure 17, and the one for recognition is given in Figure 18.

This design matrix for the regression is updated using the fuzzy weighted RLS algorithm as detailed in [207]:

$$\theta_i^k = \theta_i^{k-1} + \omega_i^k \lambda_i \overline{\boldsymbol{x}^k}\left(y^k - \overline{\boldsymbol{x}^k}^T \theta_i^{k-1}\right), \theta_i^1 = 0, \tag{270}$$

$$\omega_i^k = \omega_i^{k-1} - \frac{\lambda_i \omega_i^{k-1} \overline{\boldsymbol{x}^k}\, \overline{\boldsymbol{x}^k}^T \omega_i^{k-1^T}}{1 + \lambda_i \overline{\boldsymbol{x}^k}^T \omega_i^{k-1} \overline{\boldsymbol{x}^k}}, \omega_i^1 = \Omega I, k \in \mathbb{N} > 1, \tag{271}$$

where ω_i^k is a supplementary matrix, Ω is a positive domain-independent constant (typical value is $\Omega = 50$), I is the identity matrix.

The experimental results for this method are given in section 5.3.1.

4.2.2 TEDAPredict

The TEDAPredict structure is similar to that of TEDAClass, with the difference that it is being used for regression problems that actually means the real number of outcomes instead of the finite set members.

The fuzzy rules for TEDAPredict are defined as AnYa-type rules [111] of the following form:

$$F = \{R_i\}, i = \overline{1, N}, \tag{272}$$

$$R_i(\boldsymbol{x}): IF\ (\boldsymbol{x} \sim \boldsymbol{x}_i^*)\ THEN\ y_i = \boldsymbol{x}^T \Theta_i, \tag{273}$$

i.e. the fuzzy rule system F consists of N rules R_i, i is the rule index, $\boldsymbol{x}$ is the data sample from the data sample set $\mathfrak{X}$, $C \subset \mathbb{R}^n, F: \mathfrak{X} \to \mathrm{C}, \Theta_i$ is the design matrix for the linear regression, $\sim$ is the like predicate which shows a degree of association of the point $\boldsymbol{x}$ with the fuzzy rule. The summary of the algorithm is given in Figures 19 and 20. The experimental results for this method are given in section 5.3.2.

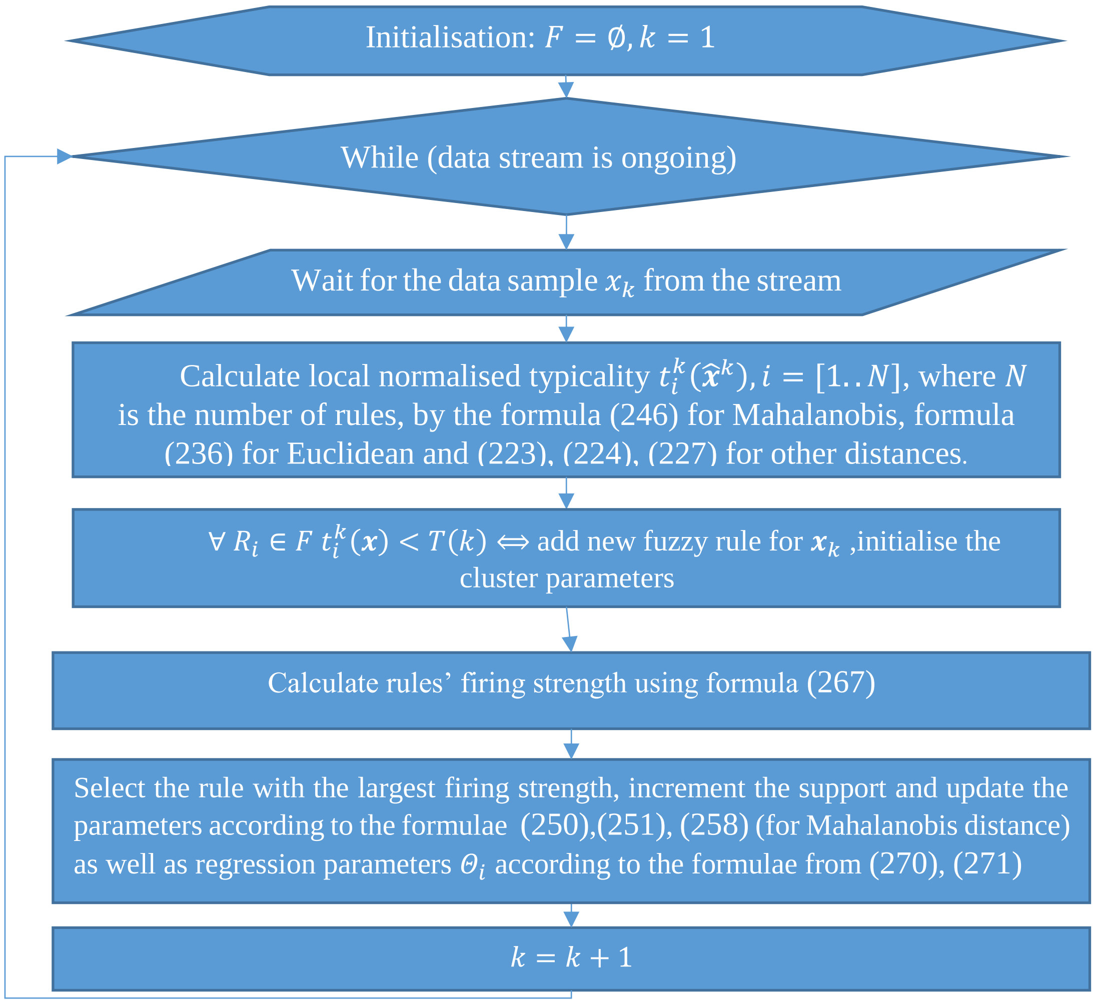

Initialisation: $F = \emptyset, k = 1$
While (data stream is ongoing)
Wait for the data sample x_k from the stream
Calculate local normalised typicality $t_i^k(\hat{\boldsymbol{x}}^k), i = [1..N]$, where N is the number of rules, by the formula (246) for Mahalanobis, formula (236) for Euclidean and (223), (224), (227) for other distances.
$\forall R_i \in F\ t_i^k(\boldsymbol{x}) < T(k) \Leftrightarrow$ add new fuzzy rule for $\boldsymbol{x}_k$,initialise the cluster parameters
Calculate rules' firing strength using formula (267)
Select the rule with the largest firing strength, increment the support and update the parameters according to the formulae (250),(251), (258) (for Mahalanobis distance) as well as regression parameters Θ_i according to the formulae from (270), (271)
$k = k + 1$

Figure 19 TEDAPredict training algorithm

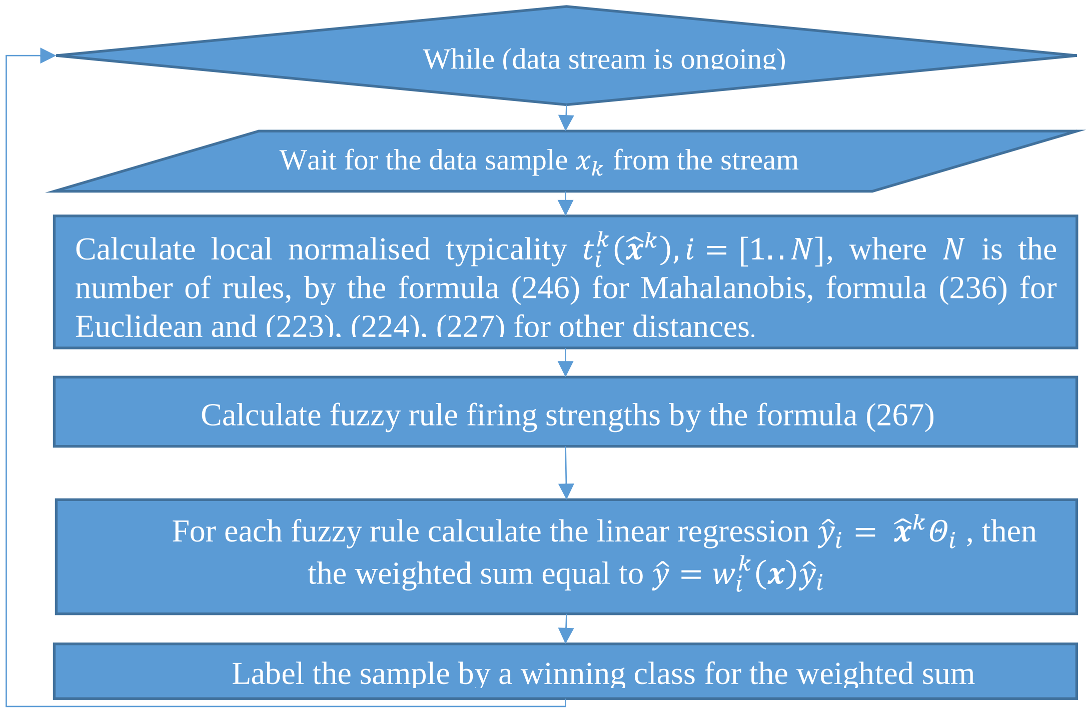

While (data stream is ongoing)
Wait for the data sample x_k from the stream
Calculate local normalised typicality $t_i^k(\hat{\boldsymbol{x}}^k), i = [1..N]$, where N is the number of rules, by the formula (246) for Mahalanobis, formula (236) for Euclidean and (223), (224), (227) for other distances.
Calculate fuzzy rule firing strengths by the formula (267)
For each fuzzy rule calculate the linear regression $\hat{y}_i = \hat{\boldsymbol{x}}^k \Theta_i$, then the weighted sum equal to $\hat{y} = w_i^k(\boldsymbol{x})\hat{y}_i$
Label the sample by a winning class for the weighted sum

Figure 20 Regression calculation algorithm

4.3 Incremental SVM classifier based on TEDA

C and ν-SVM problems, described in section 2.2.4, allow misclassification and define the upper bound of margin errors on the training set. However, the problem statement does not answer the question how to make these errors happen on less important part of the data set with importance described by some criteria, e.g. anomaly score. For many practical problems such discrimination, however, could be useful. For example, for the handwritten symbols there can be some roughly looking minority, which, by perception, should fall a victim of misclassification at the first stage. At the same time, one may not wish to misclassify accurately written symbols. To cope with such practical challenges, the SVM problem statement [18] is described here.

The idea of the method is to automatically weight the box constraints for each of the data samples according to their 'importance' within the data set. Although these weights can be defined in many ways, without loss of generality TEDA framework-based weights are described in this section to illustrate the method. These weights penalise more 'typical' support vectors for being misclassified. For each of the classes c the 'local' typicality $\tau_c^k(\boldsymbol{x})$, where $\boldsymbol{x}$ is the data sample, is defined as follows [159]:

$$\tau_c^k(\boldsymbol{x}) = 1 - 2\frac{\sum_{x_j \in X_c} d(\boldsymbol{x}, \boldsymbol{x}_j)}{\sum_{x_i \in X_c} \sum_{x_j \in X_c} d(\boldsymbol{x}_i, \boldsymbol{x}_j)}. \tag{274}$$

Here C is the class labels set, X is the data sample set, X_c is the training data sample for the class $c \in C$.

Then, for each of the classes $c \in C$, in order to enable multi-class classification, the complementary label group $\bar{c} = C \backslash c$ can be defined, consisting of all the labels but c. It allows to build the following SVM classifiers set for one versus the rest (or one versus all) model [214]:

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{n=1}^{k} C\tau_c^k(\boldsymbol{x_n})[y_n > 0]\Xi_n + \sum_{n=1}^{k} C\tau_{\bar{c}}^k(\boldsymbol{x_n})[y_n < 0]\Xi_n \rightarrow \min_{\boldsymbol{w},\Xi,\mathrm{b}} \tag{275}$$

$$t_n y(\boldsymbol{x}_n) \geq 1 - \Xi_n, \Xi_n \geq 0, n = 1 \ldots k. \tag{276}$$

The notation is the same as it was described in section 2.2.4.

One can see that the property of C- and ν-SVMs to define the upper margin errors boundary $\gamma = \frac{1}{kC}, C > 0$ is preserved, because the upper boundary $C\tau_{c_{\{m,\bar{m}\}}}^k(\boldsymbol{x}) \leq C$ is known. One can see that this alteration of the box constraints is used to 'sacrifice' the 'anomalous' data at the first stage, and, on the contrary, penalise more for misclassification of the 'typical' data. At the same time, this problem formulation preserves the initial quadratic programming solution, as only the weights have been changed. The following description is organised as follows. First, the trainable TEDA kernel is introduced (section 4.3.1). Then TEDA SVM incremental update procedure is described (section 4.3.2). The experimental results for this method are given in section 5.3.3.

4.3.1 TEDA kernel

Apart from the box constraints there is another part of the SVM problem that can be built from data, which is the kernel. In this research the TEDA kernel is proposed, which is defined as [18]

$$\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{x},\boldsymbol{y}\rangle\left(\zeta^k(\boldsymbol{x})\zeta^k(\boldsymbol{y})\right)^{\gamma}, \tag{277}$$

$$\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{x},\boldsymbol{y}\rangle\left(\frac{\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{x})\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{y})}{\left(\sum_{i=1}^k\sum_{j=1}^k d(\boldsymbol{x}_i,\boldsymbol{x}_j)\right)^2}\right)^{\gamma}, \sum_{i=1}^k\sum_{j=1}^k d(\boldsymbol{x}_i,\boldsymbol{x}_j) > 0,$$

where $\zeta^k(\boldsymbol{x})$ is a normalised data eccentricity, $\langle \boldsymbol{x},\boldsymbol{y}\rangle$ is a linear kernel, $\gamma > 0$ is a parameter. This kernel is trainable and it reflects 'eccentricity' of each of the points. It makes the anomalous points even further, and typical points closer to each other in the data space. Below it is proven that the $\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y})$ has the properties of a positive definite kernel indeed.

First, as it is described in [195], (positive definite) kernel must meet the following restrictions:

- $\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) = \ddot{\zeta}^k(\boldsymbol{y},\boldsymbol{x})$;
- $\ddot{\zeta}^k$ is non-negative definite: for Hilbert space Ω, for which kernel is defined, $\forall \boldsymbol{y}_1,\boldsymbol{y}_2,\dots\boldsymbol{y}_m \in \Omega, \forall \boldsymbol{x}_1,\boldsymbol{x}_2,\dots\boldsymbol{x}_k \in \Omega$ $M \in \mathbb{R}^{m\times m}$, $M_{ij} = \ddot{\zeta}^k(\boldsymbol{y}_i,\boldsymbol{y}_j)$ is a non-negative definite matrix. For any $\alpha \in \mathbb{R}^m$ $\alpha^T M\alpha \geq 0$.

These statements can be proven as follows (for Euclidean distance based TEDA):

1. $\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) = \ddot{\zeta}^k(\boldsymbol{y},\boldsymbol{x})$:

$$\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{x},\boldsymbol{y}\rangle\left(\frac{\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{x})\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{y})}{\left(\sum_{i=1}^k\sum_{j=1}^k d(\boldsymbol{x}_i,\boldsymbol{x}_j)\right)^2}\right)^{\gamma} =$$

$$\langle \boldsymbol{y},\boldsymbol{x}\rangle\left(\frac{\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{y})\sum_{i=1}^k d(\boldsymbol{x}_i,\boldsymbol{x})}{\left(\sum_{i=1}^k\sum_{j=1}^k d(\boldsymbol{x}_i,\boldsymbol{x}_j)\right)^2}\right)^{\gamma} = \ddot{\zeta}(\boldsymbol{y},\boldsymbol{x}).$$

2. $\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) \propto \langle \boldsymbol{x},\boldsymbol{y}\rangle\left((\|\boldsymbol{x}-\boldsymbol{\mu}^k\|^2 + [\sigma^k]^2)(\|\boldsymbol{y}-\boldsymbol{\mu}^k\|^2 + [\sigma^k]^2)\right)^{\gamma}$.

Here $\boldsymbol{\mu}^k$ is the mean and σ^k is the variance of $\{\boldsymbol{x}_1,\boldsymbol{x}_2,\dots\boldsymbol{x}_k\}$. The polynomial kernel multiplied by the linear kernel can be recognised here.

Then, the linear kernel can be replaced by any other one:

$$\ddot{\zeta}^k(\boldsymbol{x},\boldsymbol{y}) \propto K(\boldsymbol{x},\boldsymbol{y})\Big((\|\boldsymbol{x}-\boldsymbol{\mu}^k\|^2+[\sigma^k]^2)(\|\boldsymbol{y}-\boldsymbol{\mu}^k\|^2+[\sigma^k]^2)\Big)^T, \tag{278}$$

There are many ways to define trainable incrementally calculated kernel. For example, below one can see the kernel statement featuring recursive density estimation (RDE) statement [196]:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1/(1+\|\boldsymbol{x}-\boldsymbol{y}\|^2 + \Sigma_k - \|\boldsymbol{\mu}_k^2\|)\,. \tag{279}$$

One can see its equivalence to the Cauchy kernel [197] which can be proven to be a kernel:

$$D(\boldsymbol{x},\boldsymbol{y}) \propto \frac{1}{1+\|\boldsymbol{x}-\boldsymbol{y}\|^2/\alpha}, \alpha > 0. \tag{280}$$

4.3.2 TEDA SVM incremental update

The incremental SVM update method [198] is widely renowned, however it does not address the case when the box constraints and the kernel are being updated during the incremental training. The description starts with the problem definition, and then the multi-stage update procedure is proposed, which expands the original incremental SVM algorithm given in [198]. In the following description the notation used in sections 2.2.4 and 4.1.1 is maintained.

Consider training data sequence $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_k \dots\}$, $\boldsymbol{x}_k \in \Omega$. Here k is a sequential number of a data vectors. For each data vector from the sequence a label $y(\boldsymbol{x}_k)$ is assigned. It is assumed that the problem's optimal solution has been found for up to the k-th vector. Then, for the next, $(k+1)$-st, vector, the problem transforms to the following one[18]:

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{k+1} C\tau^i_{c_m}(\boldsymbol{x_i})[y_i > 0]\Xi_n + \sum_{i=1}^{k+1} C\tau^i_{\overline{c_m}}(\boldsymbol{x_i})[y_i < 0]\Xi_i \to \min_{\mathbf{w},\Xi,b} \tag{281}$$

$$t_i y(\boldsymbol{x}_i) \geq 1 - \Xi_i, \Xi_i \geq 0, i = 1 \dots k+1. \tag{282}$$

Here, $y(\boldsymbol{x})$ is expressed as

$$y(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) + b. \tag{283}$$

However, it should be accepted that $\phi(\boldsymbol{x})$ can be taken from an infinite dimensional functional space. To make the problem computationally feasible for such case and to avoid direct feature mapping estimations, the dual problem formulation can be used together with the kernel trick, where the feature mappings are replaced with kernel values $K(\boldsymbol{x}_i,\boldsymbol{x}_j) = \langle\phi^T(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)\rangle$. Technically the kernel values for the training sets can be formed into a matrix $K \in \mathbb{R}^{\mathrm{N\times N}}$, $K_{ij} = K(\boldsymbol{x}_i,\boldsymbol{x}_j)$.

The dual problem can be formulated as follows [18]:

$$\check{L}(\alpha) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i\alpha_j t_i t_j[\phi(\boldsymbol{x}_i)]^T\phi(\boldsymbol{x}_j) - \sum_{i=1}^{k}\alpha_i + b\sum_{n=1}^{k} t_i\alpha_i = \tag{284}$$

$$= \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} \alpha_i \alpha_j t_i t_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i=1}^{k} \alpha_i + b \sum_{i=1}^{k} t_i \alpha_i \to \min_{\alpha, b},$$

$$0 \le \alpha_i \le C_i, \forall i \in [1, k], \sum_{i=1}^{k} \alpha_i t_i = 0.$$

After differentiation of the Lagrangian $\check{L}(\alpha)$ Karush-Kuhn-Tucker conditions (KKT conditions) can be written as follows[18]:

$$g_j(x_j) = \frac{\partial \check{L}(\alpha)}{\partial \alpha_j} = \sum_{i=1}^{k} \alpha_i t_i t_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - 1 + t_j b = t_j y\left(\boldsymbol{x}_j\right) - 1, \tag{285}$$

$$\frac{\partial \check{L}(\alpha)}{\partial b} = \sum_{i=1}^{k} t_i \alpha_i = 0, g_j\left(x_j\right) \begin{cases} > 0, & \alpha_j = 0, \\ = 0, & 0 < \alpha_j < C_j. \\ < 0, & \alpha_j = C_j. \end{cases}$$

Based on KKT conditions, the training set Ω_L can be divided into three disjoint sets:

- margin vectors S $\left(g_j(\boldsymbol{x}_j) = 0\right)$
- error vectors E $\left(g_j(\boldsymbol{x}_j) < 0\right)$
- the rest of vectors R, which are correctly classified and are not a part of the SVM solution $\left(g_j(\boldsymbol{x}_j) > 0\right)$

After having defined the dual optimisation problem it is possible to present the sequence of the solution update in the proposed incremental SVM algorithm:

- incremental kernel matrix update (section 4.3.1)
- solution update given the new kernel matrix (section 4.3.2.2)
- solution update for the new box constraints (section 4.3.2.3)
- solution update for the new data (section 4.3.2.1)

4.3.2.1 *Addition of the new data samples*

The addition of the new samples uses a well-renowned method which was proposed in [198]. Here the derivation is given in line with the original article and the article [18] describing the method described in this thesis. Contrary to the method described in [198], it is assumed that each of the objects has its individual box constraint (which, in fact, can be also updated incrementally).

The problem is to transform the problem (284) with k data samples to the problem with the same constraints and kernel but for $k + 1$ data vectors (with one new data vector). To make the notation uncluttered, $M = k + 1$ and $Q_{ij} = t_i t_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ are defined. The following differential representation of the KKT conditions (285) is used which represents the difference between the previous and the updated KKT conditions[198], [18]:

$$\Delta g_j(\boldsymbol{x}_j) = Q_{jM}\Delta\alpha_M + \sum_{n\in S} Q_{jn}\Delta\alpha_M + t_j\Delta b, \forall\, j \in \Omega_L \cup \{M\},$$
$$0 = t_M\Delta\alpha_M + \sum_{n\in S} t_n\Delta\alpha_n. \tag{286}$$

Then, the new support vector's coefficient α_M is being changed until no further transfers between the subsets S, E, R occur. Define the following matrix:

$$\Theta = \begin{bmatrix} 0 & t_{s_1} & \cdots & t_{s_{l(S)}} \\ t_{s_1} & Q_{s_1 s_1} & \cdots & Q_{s_1 s_{l(s)}} \\ \vdots & \vdots & \ddots & \vdots \\ t_{s_{l(S)}} & Q_{s_{l(s)} s_1} & \cdots & Q_{s_{l(s)} s_{l(s)}} \end{bmatrix} \tag{287}$$

Then the margin vector set and the new vector KKT conditions can be written in the matrix form [198], [18]:

$$\Theta\begin{bmatrix}\Delta b & \Delta\alpha_{s_1} & \cdots & \Delta\alpha_{s_{l(S)}}\end{bmatrix}^T = -\begin{bmatrix}y_M & Q_{s_1 M} & \cdots & Q_{s_{l(S)} M}\end{bmatrix}^{\mathrm{T}}\Delta\alpha_M \tag{288}$$

This matrix equation can be transformed to the following system of equations [198], [18]:

$$\Delta b = \beta\Delta\alpha_M, \tag{289}$$

$$\Delta\alpha_j = \beta_j\Delta\alpha_M, \forall\, j \in D. \tag{290}$$

$$\begin{bmatrix}\beta & \beta_{s_1} & \cdots & \beta_{s_{l(s)}}\end{bmatrix}^T = -\Theta^{-1}\begin{bmatrix}y_M & Q_{s_1 M} & \cdots & Q_{s_{l(S)} M}\end{bmatrix}^T \tag{291}$$

Non-margin vectors are not included into the equation, because $\beta_n = 0\ \forall\, n \in \Omega_L \backslash S$.

Then, it is possible to write down the change of KKT conditions for each of the training set vectors depending of the newly added vector M[198], [18]:

$$\Delta g_j(x_j) = \Gamma_j\Delta\alpha_M, \forall\, j \in T\cup\{M\};$$
$$\Gamma_j = Q_{jM} + \sum_{n\in S} Q_{jn}\beta_n + t_j\beta, \forall j \notin S. \tag{292}$$

The following procedure is repeated until no further transitions between R, E, and S occurs. On each of the stages of the procedure, the maximal increment is found until one of the following conditions happen [198], [18]:

- $g_M \leq 0$, with M joining S when $g_M = 0$;
- $\alpha_M \leq 0$,with M joining E when $\alpha_M = 0$;
- $0 \leq \alpha_j \leq C_j, j \in S$ with $\alpha_j = 0$ when the j-th vector transfers from S to R, and $\alpha_j = C_j$ when transferring from S to E;
- $g_j \leq 0, \forall\, j \in E$, with $g_j = 0$ when the j-th vector transfers from E to S;
- $g_j \geq 0$, $\forall j \in R$, with $g_j = 0$ when the j-th vector transfers from R to S.

At each stage, the new support vectors must be added to the matrix Θ^{-1} in the following way[198], [18]:

$$\Theta^{-1} \leftarrow \begin{bmatrix} \Theta^{-1} & & 0 \\ & & \vdots \\ 0 & \dots & 0 \end{bmatrix} + \tag{293}$$
$$+\frac{1}{\Gamma_M}[\beta \quad \beta_{s_1} \quad \dots \quad \beta_{s_{l(s)}} \quad 1]^T[\beta \quad \beta_{s_1} \quad \dots \quad \beta_{s_{l(s)}} \quad 1].$$

One can prove that this procedure is reversible. It means that the procedure can use exclusion of the data samples from the training set (decremental learning) in a way similar to the incremental learning [198].

4.3.2.2 *Updating the kernel*

In this section, another incremental update problem is stated. Here the data set remains the same, but the kernel is modified (as a result of kernel training or just by kernel replacement). More formally, the problem

$$\breve{L}_k(\alpha) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_i\alpha_j t_i t_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i=1}^{k}\alpha_i + b\sum_{i=1}^{k} t_i\alpha_i \to \min_{\alpha,b} . \tag{294}$$

is replaced with another one[18]:

$$\breve{L}_{k+1}(\alpha) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k} \alpha_i\alpha_j t_i t_j \widehat{K}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i=1}^{k}\alpha_i + b\sum_{i=1}^{k} t_i\alpha_i \to \min_{\alpha,b}, \tag{295}$$

both with respect to the constraints (284). In consistence with the previous notation one can denote

$$Q_{ij} \;=\; t_i t_j K(\boldsymbol{x}_i, \boldsymbol{x}_j), \hat{Q}_{ij} \;=\; t_i t_j \widehat{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)\,. \tag{296}$$

$$\hat{Q}_{ij} - Q_{ij} = \Delta Q_{ij}. \tag{297}$$

After that, one can consider differential representation similar to that given in section 4.3.2.1 [18]:

$$\Delta g_j(x_j) = \sum_{n\in S}(Q_{jn} + \beta\Delta Q_{jn})\Delta\alpha_n + \beta\sum_{n\in S}\Delta Q_{jn}\alpha_n \tag{298}$$
$$+ t_j\Delta b, \sum_{n\in S} t_n\Delta\alpha_n = 0.$$

Here, $0 \le \beta \le 1$ is a linear interpolation coefficient between the old and the new kernel. In this approach, $\Delta\alpha_n$ is represented as a function of β. β is incremented gradually from 0 to 1.While the coefficient β is being incremented, the transfer conditions between margin, error and correct vectors are being consistently checked. Similarly to Θ in (287), one can define the matrix [18]

$$\breve{\Theta} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & \Delta Q_{s_1 s_1} & \dots & \Delta Q_{s_{1} s_{l_S}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \Delta Q_{s_{l_S} s_1} & \dots & \Delta Q_{s_{l_S} s_{l_S}} \end{bmatrix}. \tag{299}$$

Using this notation, the balance equation for the differential representation of the KKT conditions can be written as [18]

$$(\Theta + \beta\breve{\Theta})\left[\Delta b \quad \Delta\alpha_{s_1} \quad \dots \quad \Delta\alpha_{s_{l_S}}\right]^T = \beta\breve{\Theta}\left[b \quad \alpha_{s_1} \quad \dots \quad \alpha_{s_{l_S}}\right]^T. \tag{300}$$

and then transformed into [18]

$$\begin{aligned}\left[\Delta b \quad \Delta\alpha_{s_1} \quad \dots \quad \Delta\alpha_{s_{l_S}}\right]^T &= \beta(\Theta + \beta\breve{\Theta})^{-1}\breve{\Theta}\left[b \quad \alpha_{s_1} \quad \dots \quad \alpha_{s_{l_S}}\right]^T = \\ &= \left[b \quad \alpha_{s_1} \quad \dots \quad \alpha_{s_{l_S}}\right]^T - (\Theta + \beta\breve{\Theta})^{-1}\Theta\left[b \quad \alpha_{s_1} \quad .. \quad \alpha_{s_{l_S}}\right]^T.\end{aligned} \tag{301}$$

The update is carried out in a similar way as for the addition of the new samples in the section 4.3.2.1, but the termination condition is $\beta \in [0, 1]$.

4.3.2.3 *Updating box constraints*

Another problem of the incremental SVM update is the box constraints update. The problem is to update the solution of the problem

$$\breve{L}_k(\alpha) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i\alpha_j t_i t_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{k}\alpha_i + b\sum_{i=1}^{k} t_i\alpha_i \to \min_{\alpha, b}, \tag{302}$$

$$0 \le \alpha_i \le C_i^k, \forall i \in [1, k], \sum_{i=1}^{k}\alpha_i t_i = 0. \tag{303}$$

to the problem

$$\breve{L}_k(\alpha) = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i\alpha_j t_i t_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{k}\alpha_i + b\sum_{i=1}^{k} t_i\alpha_i \to \min_{\alpha, b}, \tag{304}$$

$$0 \le \alpha_i \le C_i^{k+1}, \forall i \in [1, k], \sum_{i=1}^{k}\alpha_i t_i = 0. \tag{305}$$

To update the solution for the new box constraints, the same incremental representation is used as for the new data in section 4.3.2.1. The previous constraints are given by the equation (285), and for the stage $(k + 1)$ the following constraints need to fulfil:

$$g_j(\boldsymbol{x}_j)\begin{cases} > 0, & \alpha_j = 0, \\ = 0, & 0 < \alpha_j < C_j^{k+1}, \\ < 0, & \alpha_j = C_j^{k+1}. \end{cases} \tag{306}$$

Initially, for each of the new constraints the fact of their violation is checked. For those constraints which are broken in the updated solution, the following algorithm is proposed. Given vector $\boldsymbol{x}_j$ for which $g_j(\boldsymbol{x}_j)$ is violated, the vector is considered as a newly added vector $\boldsymbol{x}_j$ to the k-vector SVM problem excluding vector $\boldsymbol{x}_j$. If additionally the vector belongs to the sets E or S, it is removed from the solution before the addition using decremental training technique [198]. After that, the solution repeats the incremental SVM procedure described in section 4.3.2.1.

4.3.2.4 *Incremental SVM for evolving systems*

The proposed incremental SVM solution is not restricted to a particular kernel or weighting algorithm. Additionally to the incremental SVM update algorithms, it can be successfully used in the evolving systems [18]. It can be done due to reversibility property of the incremental update procedure algorithm [198]. Using this property, one can remove the support vectors which have appeared long time ago and in this way address the shift and drift of the data stream. For example, a following sequence can be used to make the simple evolving algorithm:

- For each of the support vectors calculate within-class typicality $\tau_c^k(\boldsymbol{x})$ of a vector $\boldsymbol{x}$;
- if $\tau_c^k(\boldsymbol{x})$ is less than some pre-defined threshold, remove the support vector from the solution using decremental training procedure.

4.4 Image segmentation techniques

Additionally to the general purpose clustering, classification and regression techniques described in previous sections of this chapter, the novel image segmentation technique is proposed. This family of problems is also important for video analytics as it can be used for object detection for the subsequent stages of object tracking and/or classification. As it was discussed in section 2.2.7, image segmentation can use either general purpose data mining techniques, especially clustering methods, but also it can use the methods, specific to video analysis. In this section, such image segmentation method is proposed based on the Chan-Vese algorithm. The proposed method improves both optimisation method (two orders less iterations comparing to the original results, reported in [123] and one order less in terms of time) and functional formulation (the non-parametric version of the functional is proposed without deterioration of quality). The rest of the section is given as follows. First the optimisation scheme for the Chan-Vese functional, using the MM algorithm, is described in section 4.4.1. Then the non-parametric modification of the Chan-Vese functional is given in section 4.4.1.2. The majorant derivation for the EM algorithm is described in section 4.4.1.3. The analytical optimisation, which is carried out in order to get rid of the parameters in the non-parametric version, is given in section 4.4.1.4. The experiments with the proposed image segmentation algorithm and its comparisons with other image segmentation algorithms have been described in section 5.5.

4.4.1 Improved optimisation technique for the Chan-Vese functional

To begin with, the original Chan-Vese functional optimisation procedure [123] is briefly described to compare with the novel approach proposed in [22].

A level set function $\phi(x, y)$ is defined as

$$\phi(x,y) \begin{cases} = 0, (x,y) \in \psi, \\ > 0, (x,y) \in \xi \backslash \psi, \\ < 0, (x,y) \notin \xi. \end{cases} \tag{307}$$

where the notation is the same as for the Chan-Vese algorithm description in section 2.2.7.

The Heaviside operator $H(\phi) = H(\phi(x, y))$ is defined as follows:

$$H(\phi) = \begin{cases} 1, \phi(x,y) \geq 0, \\ 0 \text{ else.} \end{cases} \tag{308}$$

The operator discriminates between the points belonging to ξ and belonging to $\Omega \setminus \xi$, i.e. between the objects of interest and the background.

The mean intensities for the foreground and the background pixels are defined as

$$c_1(\phi) = \frac{\int_\Omega u(x,y) H\big(\phi(x,y)\big) \mathrm{d}x \, \mathrm{d}y}{\int_\Omega H\big(\phi(x,y)\big) \mathrm{d}x \, \mathrm{d}y} \tag{309}$$

and

$$c_2(\phi) = \frac{\int_\Omega u(x,y) \Big(1 - H\big(\phi(x,y)\big)\Big) \mathrm{d}x \, \mathrm{d}y}{\int_\Omega \Big(1 - H\big(\phi(x,y)\big)\Big) \mathrm{d}x \, \mathrm{d}y} \tag{310}$$

correspondingly.

Then the functions Area(ξ) and Len $(\phi(\xi))$ are defined as

$$\text{Area } (\xi) = \int_\Omega H\big(\phi(x,y)\big) \mathrm{d}x \, \mathrm{d}y \,, \tag{311}$$

$$\text{Len } (\xi) = \int_\Omega \big|\nabla H\big(\phi(x,y)\big)\big| \mathrm{d}x \, \mathrm{d}y = \int_\Omega \delta_0\big(\phi(x,y)\big) \big|\nabla H\big(\phi(x,y)\big)\big| \mathrm{d}x \, \mathrm{d}y, \tag{312}$$

where $\delta_0(\cdot)$ is a Dirac δ-function, i.e. the Heaviside function first derivative. Then the Chan-Vese functional described in the section 2.2.7 can be written as follows:

$$\begin{aligned} F(\phi) = \mu \int_\Omega \delta_0\big(\phi(x,y)\big) \big|\nabla H\big(\phi(x,y)\big)\big| \mathrm{d}x\mathrm{d}y + \nu \int_\Omega H(\phi) \mathrm{d}x\mathrm{d}y + \\ + \int_\Omega \big[\lambda_1 H(\phi) |u(\mathrm{x},\mathrm{y}) - \mathrm{c}_1(\phi))|^2 + \lambda_2 \big(1 - H(\phi)\big) |u(\mathrm{x},\mathrm{y}) - \mathrm{c}_2(\phi))|^2\big] \mathrm{d}x\mathrm{d}y. \end{aligned} \tag{313}$$

The original method [123], proposed by T.F. Chan and L.A. Vese in 2001, uses variational gradient descent. First, $c_{1,2}(\phi)$ is fixed and then Euler-Lagrange equation for ϕ is solved. To make it differentiable, the smoothened δ-function $\delta_{0\epsilon}$ is introduced. To fit it to the Euler equation, ϕ is considered as evolving in time, i.e. $\phi = \phi(x, y, t)$. The Euler-Lagrange equation is given as follows:

$$\frac{\partial \phi}{\partial t} = \delta_{0\epsilon}\big(\phi(t,x,y)\big) \times$$
$$\times \left(\mu \operatorname{div} \frac{\nabla \phi}{|\nabla \phi|} - \nu - \lambda_1 |u(x,y) - c_1(\phi)|^2 - \lambda_2 |u(x,y) - c_2(\phi)|^2\right), t > 0 \tag{314}$$

with the initial conditions

$$\phi(0,x,y) = \phi_0(x,y), (x,y) \in \Omega, \tag{315}$$

$$\frac{\delta_{0\epsilon}\phi(t,x,y)}{|\nabla\phi(t,x,y)|} \times \frac{\partial \phi}{\partial \boldsymbol{n}} = 0, (x,y) \in \partial\Omega, \tag{316}$$

where $\boldsymbol{n}$ is an exterior normal to $\partial\Omega$, and $\phi_0(x,y)$ is an initial condition. One expects it to be tending to the local minima at $t \to +\infty$. The variational gradient descent approach, which is iterative by its nature, does not provide fast convergence. The method that came out in this thesis research [22], significantly accelerates the number of iterations from several hundreds that practically prohibits the real-time applications for such methods to a few. Another problem is that the Heaviside function needs to be approximated in order to make it differentiable.

Instead of exploiting this straightforward gradient descent approach, the majorisation-minimisation (MM) algorithm [166] is used in this thesis. It improves the time performance of the algorithm by an order of magnitude by the execution time and by two orders by the iterations number as well as avoids Heaviside function approximation.

The MM algorithm [166] consists of the following two alternating steps:

- *(majorisation)* the majorant function $G_i(\phi)$ is estimated for the original, hard to optimise, function $F(\phi)$, so that $G_i(\phi) \geq F(\phi), G_i(\phi_i) = F(\phi_i)$,
- *(minimisation)* the minimisation procedure is carried out for the function $G_i(\phi_i)$, so that $\phi_{i+1} = \arg\min_{\phi} \; G_i(\phi)$.

These conditions are used for the proof of the MM algorithm convergence [166], because on each step the majorant coincides with the original function, and the lower bound of the majorant is given by the function under optimisation itself.

For the first stage, some initial guess ϕ_1 is used. The majorisation and minimisation procedures are repeated until the accomplishment of the convergence condition, e.g. $|F(\phi_{i+1}) - F(\phi_i)| < \epsilon$, where ϵ is some pre-defined constant.

The description of the method follows hereafter. First, the following function is defined [22]:

$$\Xi\colon \Omega \to \{0,1\}, \Xi(x,y) = H\big(\phi(x,y)\big) \tag{317}$$

Then, the functional can be rewritten as

$$\begin{aligned} F(\phi) = \mu \int_\Omega \delta_0\big(\phi(x,y)\big)|\nabla\Xi(x,y)|\mathrm{d}x\mathrm{d}y + \nu \int_\Omega \Xi(x,y)\mathrm{d}x\mathrm{d}y + \\ +\lambda_1 \int_\Omega \Xi(x,y)|\mathrm{u}(\mathrm{x},\mathrm{y}) - \mathrm{c}_1(\Xi)|^2\mathrm{d}x\mathrm{d}y + \\ +\lambda_2 \int_\Omega \big(1-\Xi(x,y)\big)|\mathrm{u}(\mathrm{x},\mathrm{y}) - \mathrm{c}_2(\Xi)|^2\mathrm{d}x\mathrm{d}y. \end{aligned} \tag{318}$$

Here, it should be emphasised that the Heaviside function is not approximated in the proposed solution [22]. As the solution is iterative, it is proposed to define abbreviations $c_1 = c_1(\Xi_k)$ and $c_2 = c_2(\Xi_k)$ where Ξ_k is the function Ξ at the Ξ-th stage of the optimisation.

Using the results, derived in the section 4.4.1.3, the majorant can be expressed as

$$G_k^{x,y}(\Xi, c_{1,2}) = \begin{cases} \mu\left(|\nabla\Xi_k| + \dfrac{|\nabla\Xi|^2 - |\nabla\Xi_k|^2}{2|\nabla\Xi_k| + \epsilon}\right) + \lambda_2(u-c_2)^2 + \\ \quad +\lambda_1(\Xi - \Xi_k)\max\limits_q(u-q)^2 + \nu\Xi, \Xi_k = 0, \\ \mu\left(|\nabla\Xi_k| + \dfrac{|\nabla\Xi|^2 - |\nabla\Xi_k|^2}{2|\nabla\Xi_k| + \epsilon}\right) + \lambda_1(u-c_1)^2 + \\ \quad +\lambda_2(\Xi_\mathrm{k} - \Xi)\max\limits_q(u-q)^2 + \nu\Xi, \Xi_k = 1. \end{cases} \tag{319}$$

Here, $u = u(x,y)$, Ξ_k is an abbreviation of $\Xi_k(x,y)$, and Ξ of $\Xi(x,y)$.

4.4.1.1 *Graph cut optimisation*

After the description of the MM algorithm scheme, the optimisation itself has to be described. First, the graph cut problem is formulated as it is stated for the Boykov-Kolmogorov algorithm [164], and then it is shown how to fit the majorant (319) to meet the energy functional optimisation problem restrictions.

The energy functional optimisation problem is formulated as follows [164]:

$$E(\Xi) = \sum_{(x,y)\in\Omega} D(\Xi(x,y),x,y) + \tag{320}$$
$$+ \sum_{\substack{(x,y)\in\Omega,\\ (x_1,y_1)\in \mathrm{N}(x,y)}} V(\Xi(x,y),\Xi(x_1,y_1),x,y,x_1,y_1).$$

Here, the function $D(\Xi(x,y),x,y)$ is referred to as an unary potential, $V(\Xi(x,y),\Xi(x_1,y_1),x,y,x_1,y_1)$ is an interactional (pairwise) potential, and $\mathrm{N}(x,y)$ is a set of neighbours of the point (x,y). Ξ should be understood, in contrary to the previous section, as some binary labelling function $\Omega \times \Omega \to \{0,1\}$ (however, it has the same notation because it is used for finding actual labelling for the given Chan-Vese problem).

Additionally to this, the following submodularity condition is imposed:

$$\forall (x,y) \in \Omega, (x_1,y_1) \in N\big((x,y)\big), V\,(0,1,x,y,x_1,y_1) + V(1,0,x,y,x_1,y_1) \tag{321}$$
$$\geq V(1,1,x,y,x_1,y_1) + V(0,0,x,y,x_1,y_1).$$

It ensures non-negative weights in the graph cut algorithm when proving its correctness and can be interpreted as an analogue of the convexity property for a discrete case.

It is shown in the work [22] that the optimisation problem can be solved using graph cut for the weighted graph. The graph can be built for the image segmentation problem as follows:

- the following nodes are defined in the graph: for each pixel the node is defined, and there are also two special terminal nodes, which are referred to as source, s-node (corresponding to "0"-label) and sink, t-node (corresponding to "1"-label)
- the graph connections are defined between each of the pixel nodes and their neighbours' nodes (n-links) and between pixel nodes and terminal nodes (t-links)

In order to minimise the function (320), the graph cut algorithm cuts the graph between the zones of influence of the s and t nodes which actually defines a segmentation of the image corresponding to the graph. The two node subsets are denoted here as S and T, where s-node $s \in S$, and $t \in T$. It is assumed that the edges adjoining the subset nodes are included into the subgraphs. The sum of the weights of the edges on the border between S and T constitute the cost of the graph cut.

Then, it is needed to relate the original problem of Chan-Vese functional optimisation to the graph cut optimisation. The functional under optimisation is defined as [22]

$$G = \sum_{(x,y)\in\Omega} G_k^{(x,y)}(\Xi(x,y),c_1,c_2) \to \min_{\Xi,c_1,c_2} \tag{322}$$

and can be structured as follows [22]:

$$G_k^{(x,y)} = a(\Xi_k, c, x, y) + b(\Xi, \Xi_k, x, y). \tag{323}$$

Here [22]

$$a(\Xi_k, c, x, y) = \begin{cases} \frac{1}{2}\mu|\nabla\Xi_k(x,y)| + \lambda_2(u(x,y) - c_2)^2, \text{if } \Xi_k(x,y) = 0, \\ \frac{1}{2}\mu|\nabla\Xi_k(x,y)| + \lambda_1(u(x,y) - c_1)^2, \text{if } \Xi_k(x,y) = 1; \end{cases} \tag{324}$$

$$b(\Xi_k, \Xi, x, y) =$$

$$= \begin{cases} \frac{\mu|\nabla\Xi_k(x,y)|^2}{2(|\nabla\Xi_k(x,y)| + \epsilon)} + \nu\Xi(x,y) + \lambda_1\Xi(x,y)\max_q(u(x,y) - q)^2 , \\ \text{if } \Xi_k(x,y) = 0, \\ \frac{\mu|\nabla\Xi_k(x,y)|^2}{2(|\nabla\Xi_k(x,y)| + \epsilon)} + \nu\Xi(x,y) + \lambda_2\big(1 - \Xi(x,y)\big)\max_q(u(x,y) - q)^2 , \\ \text{if } \Xi_k(x,y) = 1. \end{cases} \tag{325}$$

One can see that the function $a(\Xi_k, c, x, y)$ can be optimised analytically with respect to $c_{1,2}$, while $b(\Xi_k, \Xi, x, y)$ depends on Ξ but not on $c_{1,2}$. The optimisation of the function $\sum_{(x,y)\in\Omega} b(\Xi_k, \Xi, x, y)$ can be performed using the graph cuts method that is described hereafter.

To convert $\sum_{(x,y)\in\Omega} b(\Xi_k, \Xi, x, y)$ to the graph cut optimisation problem, one needs to divide the expression on the unary and pairwise potentials.

Unary potentials are defined by the following expression [22]:

$$D\,(\Xi, \Xi_k, x, y) = \begin{cases} \nu\Xi(x,y) + \lambda_1\Xi(x,y)\max_{\mathrm{q}}(u(x,y) - q)^2, \Xi_k(x,y) = 0, \\ \nu\Xi(x,y) + \lambda_2\big(1 - \Xi(x,y)\big)\max_{\mathrm{q}}(u(x,y) - q)^2, \Xi_k(x,y) = 1. \end{cases} \tag{326}$$

The pairwise potential can be written as [22]

$$\begin{gathered} \hat{V}(\Xi, \Xi_k, x, y) = \frac{\mu|\nabla\Xi_k(x,y)|^2}{2\big(\big|\nabla\Xi_{k(x,y)}\big| + \epsilon\big)} = \frac{\mu|\nabla\Xi_k(x,y)|^2}{2(|\nabla\Xi_k(x,y)| + \epsilon)} \times \\ \times \Big[\big(\Xi(x,y) - \Xi(x,y-1)\big)^2 + \big(\Xi(x,y) - \Xi(x-1,y)\big)^2\Big] = \end{gathered} \tag{327}$$

$$= f(\mu, x, y)I\big(\Xi(x,y) \neq \Xi(x-1,y)\big) + f(\mu, x, y)I\big(\Xi(x,y) \neq \Xi(x,y-1)\big),$$

where $I(\cdot)$ is a predicate, equal to one, if the argument is true, and equal to zero otherwise, and

$$f(\mu, x, y) = \frac{\mu|\nabla\Xi_k(x,y)|^2}{2\big(\big|\nabla\Xi_{k(x,y)}\big| + \epsilon\big)}. \tag{328}$$

After substitution of the formulae below $\sum_{(x,y)\in\Omega} b(\Xi_k, \Xi, x, y)$ can be represented as [22]

$$\sum_{(x,y)\in\Omega} b(\Xi_k,\Xi,x,y) = \sum_{(x,y)\in\Omega}\left[\hat{V}(\Xi,\Xi_k,x,y) + D(\Xi,\Xi_k,x,y)\right] =$$
$$= \sum_{(x,y)\in\Omega}\left[f(\mu,x,y)\left(I^{\left(\Xi^{(x,y)}\neq\Xi(x-1,y)\right)} + I^{\left(\Xi(x,y)\neq\Xi(x,y-1)\right)}\right)\right] +$$
$$+ \sum_{(x,y)\in\Omega} D(\Xi,\Xi_k,x,y) =$$
$$= \left\{V'(\mu,x,y,x_1,y_1) \coloneqq f(\mu,x,y)I\big(\Xi(x,y)\neq\Xi(x_1,y_1)\big)\right\} = \tag{329}$$
$$= \sum_{(x,y)\in\Omega}\left[V'(\mu,x,y,x-1,y) + V'(\mu,x,y,x,y-1) + D(\Xi,\Xi_k,x,y)\right] =$$
$$= \frac{1}{2}\sum_{\substack{(x,y)\in\Omega,\\(x_1,y_1)\in \mathrm{N}(x,y)}} V'(\mu,x,y,x_1,y_1) + \sum_{(x,y)\in\Omega} D(\Xi,\Xi_k,x,y).$$

Here $I^{(x)} = I(x)$ was introduced to make the notation more compact. To make the notation uncluttered, it is worth denoting

$$V(\mu,x,y,x_1,y_1) = \frac{1}{2}V'(\mu,x,y,x_1,y_1). \tag{330}$$

Then all the components of the graph cut optimisation problem are defined, and the functional can be optimised. However, for the practical reasons, to speed up convergence, it is worth modifying the unary potential to tie it with the values of c_1 and c_2 from the previous optimisation step [22]:

$$\hat{D}(\Xi,\Xi_k,x,y)$$
$$= \begin{cases} \nu\Xi(x,y) + \lambda_1\Xi(x,y)\max\limits_{q}(u(x,y)-q)^2 + \left(u(x,y)-c_1^k\right)^2, \\ \qquad \text{if } \Xi_k(x,y) = 0, \\ \nu\Xi(x,y) + \lambda_2\big(1-\Xi(x,y)\big)\max\limits_{q}(u(x,y)-q)^2 + \left(u(x,y)-c_2^k\right)^2, \\ \qquad \text{if } \Xi_k(x,y) = 1, \end{cases} \tag{331}$$

where $c_{1,2}^k$ are the values of $c_{1,2}$ from the previous iteration of the majorisation-minimisation algorithm. These additional terms $\left(u(x,y)-c_1^k\right)^2$, $\left(u(x,y)-c_2^k\right)^2$ are non-negative, hence it does not break the majorisation condition.

One can see that this optimisation problem formulation reveals the similarity between the Chan-Vese and MRF (Markov random fields) image segmentation algorithms [135]. In MRF, the problem is formulated as aiming to the most preferable hidden variables configuration (i.e. segmentation labels). The probability is formulated via visible variables (i.e.

red, green and blue intensities of the image) with some model parameters. In both, Chan-Vese and MRF, cases the problem is representable by graphs however the Chan-Vese algorithm also uses unary functionals additionally to the pairwise ones stated for the MRF model.

4.4.1.2 *Non-parametric Chan-Vese functional*

However, the improvement of the optimisation procedure is not the only contribution of this thesis to the Chan-Vese functional segmentation. In this section the non-parametric functional is defined complimentary to the original, parametric, version of the functional, described in the previous subsections. First, the parameterisation is removed, if $\lambda_{1,2}$ are included into the optimisation process [22]:

$$\begin{gathered}\tilde{F}\,(\Xi, U, c_1, c_2, \lambda_1, \lambda_2) = \mu \int_\Omega |\nabla \Xi(x,y)|\, \mathrm{d}x\, \mathrm{d}y + \\ +\lambda_1 \int_\Omega \Xi(x,y)|\mathrm{u}(x,y) - c_1|^2 \mathrm{d}x\, \mathrm{d}y + \\ +\lambda_2 \int_\Omega \big(1 - \Xi(x,y)\big)|\mathrm{u}(x,y) - c_2|^2 \mathrm{d}x\, \mathrm{d}y.\end{gathered} \tag{332}$$

Note that $\mu > 0$ remains fixed because otherwise the optimisation will be straightforward. The division on μ leads to the equivalent problem:

$$\begin{gathered}F\,(\Xi, U, c_1, c_2, \lambda_1, \lambda_2) = \int_\Omega |\nabla \Xi(x,y)|\, \mathrm{d}x\, \mathrm{d}y + \\ +\frac{\lambda_1}{\mu} \int_\Omega \Xi(x,y)|\mathrm{u}(x,y) - c_1|^2 \mathrm{d}x\, \mathrm{d}y + \\ +\frac{\lambda_2}{\mu} \int_\Omega \big(1 - \Xi(x,y)\big)|\mathrm{u}(x,y) - c_2|^2 \mathrm{d}x\, \mathrm{d}y.\end{gathered} \tag{333}$$

Therefore $\mu = 1$ can be assumed hereafter without loss of generalisation.

The next step is to define Gibbs distribution, which helps to exploit Hammersley-Clifford theorem [165] stating that the Gibbs distribution statement is equivalent to the MRF statement. First, consider the following Gibbs probability distribution function [22]:

$$P\left(\Xi, U \middle| c, \lambda_{1,2}\right) = \frac{\exp\left(-\frac{1}{2} F\,(\Xi, U, c_1, c_2, \lambda_1, \lambda_2)\right)}{\tilde{Z}(c, \lambda_1, \lambda_2)}, \tag{334}$$

where $\tilde{Z}(c, \lambda_1, \lambda_2)$ is the normalisation condition given by the expression [22]

$$\int\!\!\int \frac{\exp\left(-\frac{1}{2} F\,(\Xi, U, c_1, c_2, \lambda_1, \lambda_2)\right)}{\tilde{Z}(c, \lambda_1, \lambda_2)} \mathrm{d}U\, \mathrm{d}\Xi = 1. \tag{335}$$

Here, one can notice that the function $F(\Xi, U)$ is actually discrete hence the integral can be replaced by a summation. However, the integrals are used in this section to keep the notation simple.

Substituting, one can obtain the following expression for $P\left(\Xi, U\middle|c, \lambda_{1,2}\right)$[22]:

$$P(\Xi, U|c, \lambda_1, \lambda_2) = \exp\left[\frac{\left(-\frac{1}{2}\sum_{(x,y)\in\Omega}|\nabla\Xi_k(x,y)|\right)}{\tilde{Z}(c, \lambda_1, \lambda_2)}\right] \times$$
$$\times \prod_{\substack{(x,y)\in\Omega,\\ \Xi(x,y)=1}} \frac{\lambda_1}{2}\exp\left(-\frac{1}{2}\lambda_1|u(x,y) - c_1|^2\right) \times \tag{336}$$
$$\times \prod_{\substack{(x,y)\in\Omega,\\ \Xi(x,y)=0}} \frac{\lambda_2}{2}\exp\left(-\frac{1}{2}\lambda_2|u(x,y) - c_2|^2\right).$$

After scaling the normalisation coefficient $Z(c, \lambda_1, \lambda_2) \propto \tilde{Z}(c, \lambda_1, \lambda_2)$ one can obtain [22]

$$P(\Xi, U|c, \lambda_1, \lambda_2) = \exp\left[\frac{\left(-\frac{1}{2}\sum_{(x,y)\in\Omega}|\nabla\Xi_k(x,y)|\right)}{Z(c, \lambda_1, \lambda_2)}\right] \times$$
$$\times \prod_{\substack{(x,y)\in\Omega,\\ \Xi(x,y)=1}} \sqrt{\frac{\lambda_1}{2\pi}}\exp\left(-\frac{1}{2}\lambda_1|u(x,y) - c_1|^2\right) \times \tag{337}$$
$$\times \prod_{\substack{(x,y)\in\Omega,\\ \Xi(x,y)=0}} \sqrt{\frac{\lambda_2}{2\pi}}\exp\left(-\frac{1}{2}\lambda_2|u(x,y) - c_2|^2\right).$$

Assuming that $u \in (-\infty, +\infty)$, one can integrate out u [22]:

$$P(\Xi|c, \lambda_1, \lambda_2) = \frac{\exp\left(-\frac{1}{2}\sum_{(x,y)\in\Omega}|\nabla\Xi_k(x,y)|\right)}{Z(c, \lambda_1, \lambda_2)}. \tag{338}$$

It can be noticed that the exponent does not depend neither on c nor on λ_1, λ_2. Therefore, the normalisation coefficient Z does not depend on c as well. It means that the original functional optimisation problem is equivalent to the following one, when $\lambda_{1,2}$ are fixed:

$$P(\Xi, U|c, \lambda_1, \lambda_2) \to \max_{\Xi, c}\ . \tag{339}$$

After the analytical optimisation procedure, described in section 4.4.1.4, one can obtain [22]

$$c_1^*(\Xi) = \frac{\int_\Omega \Xi(x,y)u(x,y)\mathrm{d}x\mathrm{d}y}{\int_\Omega \Xi(x,y)\mathrm{d}x\,\mathrm{d}y}, \tag{340}$$

$$\lambda_1^*(\Xi, \mathrm{c}_1) = \frac{\int_\Omega \Xi(x,y)\mathrm{d}x\mathrm{d}y}{\int_\Omega \Xi(x,y)\,|u(x,y) - c_1|^2\,\mathrm{d}x\,\mathrm{d}y}, \tag{341}$$

$$c_2^*(\Xi) = \frac{\int_\Omega \big(1 - \Xi(x,y)\big)u(x,y)\mathrm{d}x\mathrm{d}y}{\int_\Omega \big(1 - \Xi(x,y)\big)\mathrm{d}x\,\mathrm{d}y}, \tag{342}$$

$$\lambda_2^*(\Xi, \mathrm{c}_2) = \frac{\int_\Omega \big(1 - \Xi(x,y)\big)\mathrm{d}x\mathrm{d}y}{\int_\Omega \big(1 - \Xi(x,y)\big)\,|u(x,y) - c_2|^2\,\mathrm{d}x\,\mathrm{d}y}. \tag{343}$$

After that one can substitute and obtain [22]

$$\begin{gathered} G\big(\Xi, U, c(\Xi), \lambda(\Xi)\big) = \\ = \int_\Omega [|\nabla\Xi(x,y)| + \Xi(x,y)]\mathrm{d}x\,\mathrm{d}y + \int_\Omega \big(1 - \Xi(x,y)\big)\mathrm{d}x\,\mathrm{d}y + \\ + \int_\Omega \log\left(\frac{1}{\lambda_1(\Xi)}\right)\Xi(x,y)\,\mathrm{d}x\,\mathrm{d}y + \\ + \int_\Omega \log\left(\frac{1}{\lambda_2(\Xi)}\right)\big(1 - \Xi(x,y)\big)\mathrm{d}x\,\mathrm{d}y. \end{gathered} \tag{344}$$

Then, it is possible to consider a constrained case $\lambda_1 = \lambda_2 = \lambda$. Using the analytical optimisation similar to that in the section 4.4.1.4, it can be found that [22]

$$\begin{gathered} \lambda^*(\Xi, \mathrm{c}_1, \mathrm{c}_2) = |\Omega|\left[\int_\Omega \Xi(x,y)|u(x,y) - c_1|^2 + \right. \\ \left. + \big(1 - \Xi(x,y)\big)|u(x,y) - c_2|^2\mathrm{d}x\,\mathrm{d}y\right]^{-1}. \end{gathered} \tag{345}$$

The part of the majorant for the MM algorithm is calculated in the following way [22]:

$$\begin{gathered} \int_\Omega \Xi(x,y)\,\mathrm{d}x\,\mathrm{d}y \times \log\left[\frac{1}{\lambda(\Xi)}\right] = \int_\Omega \Xi(x,y)\mathrm{d}x\,\mathrm{d}y \times \\ \times \log\left[\frac{\int_\Omega \Xi(x,y)|u(x,y) - c_1(\Xi)|^2\mathrm{d}x\,\mathrm{d}y}{\int_\Omega \Xi(x,y)\mathrm{d}x\,\mathrm{d}y}\right] \le \\ \le \{\text{due to log function concavity}\} \le \\ \le \int_\Omega \Xi(x,y)\,\mathrm{d}x\,\mathrm{d}y \times \log\left[\frac{1}{\lambda(\Xi_\mathrm{k})}\right] - 1 + \\ + \lambda(\Xi_k) \times \frac{\int_\Omega \Xi(x,y)|u(x,y) - c_1(\Xi)|^2\mathrm{d}x\,\mathrm{d}y}{\int_\Omega \Xi(x,y)\mathrm{d}x\,\mathrm{d}y}. \end{gathered} \tag{346}$$

After that, all other summands of the majorant are substituted similarly to the expression (319). After majorisation it is possible to employ the graph cut algorithm for the majorant optimisation.

4.4.1.3 *The majorant derivation*

According to the majorisation-minimisation (MM) algorithm [166], the majorant is defined by the following equations:

$$G_k(\Theta_k) = F(\Theta_k), \tag{347}$$

$$G_k(\Theta) \geq F(\Theta)\ \forall\ \Theta. \tag{348}$$

The majorant is composed of several summands, each fulfilling the following conditions. First, the following functions are defined for majorisation[22]:

$$F_1(\Xi, c_1) = \Xi(x,y)|u(x,y) - c_1^*(\Xi)|^2; \tag{349}$$

$$F_2(\Xi, c_2) = \big(1 - \Xi(x,y)\big)|u(x,y) - c_2^*(\Xi)|^2; \tag{350}$$

$$F_3\big(\Xi(x,y)\big) = |\nabla\Xi(x,y)|. \tag{351}$$

For $F_1(\Xi, c_1)$ the aim is to build the majorant with $G_{1k}(\Xi_k, c_1) = F_1(\Theta_k, c_1)$. In the case of $\Xi_k(x,y) = 0$, this function can be stated as [22]:

$$\begin{gathered} F_1(\Xi, c_1) = \Xi(x,y)|u(x,y) - c_1|^2 = \\ = \Xi_k(x,y)|u(x,y) - c_1|^2 + \big(\Xi(x,y) - \Xi_k(x,y)\big)|u(x,y) - c_1|^2 \leq \\ \leq \Xi_k(x,y)|u(x,y) - c_1|^2 + \big(\Xi(x,y) - \Xi_k(x,y)\big)\max_{\mathrm{q}}(u(x,y) - q)^2 = \\ = G_{1k}(\Xi, c_1). \end{gathered} \tag{352}$$

This result exploits the fact that $\Xi(x,y) - \Xi_k(x,y) \geq 0$.

Then, if $\Xi_{k(x,y)} = 1$, the majorant can be represented as [22]

$$\begin{gathered} F_1(\Xi, c_1) = \Xi(x,y)|u(x,y) - c_1|^2 = \Xi_k(x,y)|u(x,y) - c_1|^2 + \\ +\big(\Xi(x,y) - \Xi_k(x,y)\big)|u(x,y) - c_1|^2 \leq \Xi_k(x,y)|u(x,y) - c_1|^2 \\ = G_{1k}(\Xi, c_1). \end{gathered} \tag{353}$$

For the function $F_2(\Xi_k, c_2)$, the majorant looks as follows. If $\Xi_k(x,y) = 0$ then [22]

$$F_2(\Xi, c_2) = \big(1 - \Xi(x,y)\big)|u(x,y) - c_2|^2 = \tag{354}$$

$$= \left(1 - \Xi_k(x,y)\right)|u(x,y) - c_2|^2 + \left(\Xi_k(x,y) - \Xi(x,y)\right)|u(x,y) - c_2|^2 \leq$$

$$\leq \left(1 - \Xi_k(x,y)\right) \max_q |u(x,y) - c_2|^2 = G_{2k}(\Xi, c_2)$$

(here, the fact that $\Xi(x,y) - \Xi_k(x,y) \geq 0$ is used).

If $\Xi_k(x,y) = 1$, then

$$F_2(\Xi, c_2) = \left(1 - \Xi(x,y)\right)|u(x,y) - c_2|^2 =$$

$$= \left(1 - \Xi_k(x,y)\right)|u(x,y) - c_2|^2 + \left(\Xi_k(x,y) - \Xi(x,y)\right)|u(x,y) - c_2|^2 \leq$$

$$\leq \left(1 - \Xi_k(x,y)\right)|u(x,y) - c_2|^2 + \qquad (355)$$

$$+\left(\Xi_k(x,y) - \Xi(x,y)\right) \max_q |u(x,y) - q|^2 =$$

$$= G_{2k}(\Xi, c_2).$$

For $F_{3k}(\Xi)$, the following majorant is used, which exploits the concavity property of the square root [22]:

$$F_{3k}(\Xi) \leq |\nabla\Xi_k(x,y)| + \frac{|\nabla\Xi(x,y)|^2 - |\nabla\Xi_k(x,y)|^2}{2|\nabla\Xi_k(x,y)|} = \hat{G}_{3k}(\Xi). \qquad (356)$$

To avoid issues with zero gradient, the majorant should be modified as follows [22]:

$$G_{3k}(\Xi) = |\nabla\Xi_k(x,y)| + \frac{|\nabla\Xi(x,y)|^2 - |\nabla\Xi_k(x,y)|^2}{2|\nabla\Xi_k(x,y)| + \epsilon}, \epsilon > 0, \epsilon \leq \frac{\sqrt{2} - 1}{2}. \qquad (357)$$

This result exploits the fact that the gradient can take one of three values: $\{0, 1, \sqrt{2}\}$, and one can see that this correction does not break the majorisation conditions.

After assembling all the majorant components together one can obtain [22]

$$G_k^{(x,y)}(\Xi, c_1, c_2) = \mu\left(\left|\nabla\Xi_k^{(x,y)}\right| + \frac{\left|\nabla\Xi^{(x,y)}\right|^2 - \left|\nabla\Xi_k^{(x,y)}\right|^2}{2\left|\nabla\Xi_k^{(x,y)}\right| + \epsilon}\right) + \nu\Xi^{(x,y)} + \qquad (358)$$

$$+\begin{cases} \lambda_1\left(\Xi_k^{(x,y)}|u(x,y)-c_1|^2+\left(\Xi^{(x,y)}-\Xi_k^{(x,y)}\right)\max_{\mathrm{q}}|u(x,y)-q|^2\right)+ \\ +\lambda_2\left(1-\Xi_k^{(x,y)}\right)|u(x,y)-c_2\,|^2, \text{if } \Xi_k^{(x,y)}=0, \\ \lambda_1\Xi_k^{(x,y)}|u(x,y)-c_1|^2+ \\ +\lambda_2\left(\left(1-\Xi_k^{(x,y)}\right)|u(x,y)-c_2|^2+\left(\Xi_k^{(x,y)}-\Xi^{(x,y)}\right)\max|u(x,y)-q|^2\right), \\ \text{if } \Xi_k^{(x,y)}=1. \end{cases}$$

Here $\Xi_k^{(x,y)}=\Xi_k(x,y), \Xi^{(x,y)}=\Xi(x,y)$.

The final expression, after exclusion of the zero terms, is [22]

$$G_k^{(x,y)}(\Xi,c_1,c_2)=\mu\left(\left|\nabla\Xi_k^{(x,y)}\right|+\frac{\left|\nabla\Xi^{(x,y)}\right|^2-\left|\nabla\Xi_k^{(x,y)}\right|^2}{2|\nabla\Xi_k(x,y)|+\epsilon}\right)+\nu\Xi^{(x,y)}+$$
$$+\begin{cases} \lambda_1\Xi^{(x,y)}\max_{\mathrm{q}}|u(x,y)-q|^2+\lambda_2|u(x,y)-c_2\,|^2, \text{if } \Xi_k^{(x,y)}=0, \\ \lambda_1|u(x,y)-c_1|^2+ \\ +\lambda_2\left(\left(1-\Xi^{(x,y)}\right)\max|u(x,y)-q|^2\right), \text{if } \Xi_k^{(x,y)}=1. \end{cases} \tag{359}$$

4.4.1.4 *Analytical optimisation of the modified Chan-Vese functional coefficients*

The problem considered here is to perform analytical optimisation of the function

$$\begin{aligned} P(\Xi,U|c_1,c_2,\lambda_1,\lambda_2) &= \frac{1}{Z(c_1,c_2,\lambda_1,\lambda_2)}\times\prod_{(x,y)\in\Omega}\sqrt{\frac{1}{2\pi}}\times \\ &\times\prod_{\substack{(x,y)\in\Omega,\\ \Xi(x,y)=1}}\exp\left(-\frac{1}{2}(\lambda_1|u(x,y)-c_1|^2-\log(\lambda_1))\right)\times \\ &\times\prod_{\substack{(x,y)\in\Omega,\\ \Xi(x,y)=1}}\exp\left(-\frac{1}{2}(\lambda_2|u(x,y)-c_2|^2-\log(\lambda_2))\right)\times \\ &\times\exp\left(-\frac{1}{2}\sum_{(x,y)\in\Omega}|\nabla\Xi(x,y)|\right)\to\max_{c_1,c_2,\lambda_1,\lambda_2}. \end{aligned} \tag{360}$$

Then, after taking logarithm and removing constants, one can obtain [22]

$$G\left(\Xi, U, c_{1,2}, \lambda_{1,2}\right) = \int_\Omega |\nabla \Xi(x,y)| \mathrm{d}x\,\mathrm{d}y + \lambda_1 \int_\Omega \Xi(x,y)|u(x,y) - c_1|^2 \mathrm{d}x\,\mathrm{d}y + \\ +\lambda_2 \int_\Omega (1 - \Xi(\mathrm{x},\mathrm{y}))|u(x,y) - c_2|^2 \mathrm{d}x\,\mathrm{d}y - \int_\Omega \Xi(x,y)\mathrm{d}x\,\mathrm{d}y \times \log(\lambda_1) - \\ - \int_\Omega \left(1 - \Xi(x,y)\right)\mathrm{d}x\,\mathrm{d}y \times \log(\lambda_2) \to \min_{c_{1,2},\lambda_{1,2}}. \tag{361}$$

In order to perform analytical optimisation of $G(\Xi, U, c, \lambda)$, the zero crosses for the derivatives need to be analysed [22]:

$$\frac{\partial G\left(\Xi, U, c_{1,2}, \lambda_{1,2}\right)}{\partial c_1} = 2\lambda_1 \int_\Omega \Xi(x,y)(u(x,y) - c_1)\mathrm{d}x\,\mathrm{d}y, \tag{362}$$

$$\frac{\partial G\left(\Xi, U, c_{1,2}, \lambda_{1,2}\right)}{\partial \lambda_1} = \int_\Omega \Xi(x,y)|u(x,y) - c_1|^2 \mathrm{d}x\,\mathrm{d}y - \frac{1}{\lambda_1}\int_\Omega \Xi(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{363}$$

The final solutions are written as follows [22]:

$$c_1^*(\Xi) = \frac{\int_\Omega \Xi(x,y)u(x,y)\mathrm{d}x\mathrm{d}y}{\int_\Omega \Xi(x,y)\mathrm{d}x\,\mathrm{d}y}, \tag{364}$$

$$\lambda_1^*(\Xi, \mathrm{c}_1) = \frac{\int_\Omega \Xi(x,y)\mathrm{d}x\mathrm{d}y}{\int_\Omega \Xi(x,y)\;|u(x,y) - c_1|^2\;\mathrm{d}x\;\mathrm{d}y}, \tag{365}$$

$$c_2^*(\Xi) = \frac{\int_\Omega \left(1 - \Xi(x,y)\right)u(x,y)\mathrm{d}x\mathrm{d}y}{\int_\Omega \left(1 - \Xi(x,y)\right)\mathrm{d}x\,\mathrm{d}y}, \tag{366}$$

$$\lambda_2^*(\Xi, \mathrm{c}_2) = \frac{\int_\Omega \left(1 - \Xi(x,y)\right)\mathrm{d}x\mathrm{d}y}{\int_\Omega \left(1 - \Xi(x,y)\right)\;|u(x,y) - c_2|^2\;\mathrm{d}x\;\mathrm{d}y}. \tag{367}$$

4.5 Big Data versions of the TEDA-based clustering and classification algorithms

Although not the part of the initial aims of the thesis, the idea to apply the proposed classifiers to the Big Data applications has been tried in order to demonstrate the applicability of the TEDA-based methods to the practical problems. This concept has a significant relevance to the general idea of this thesis, as the method can be used for distributed computation for pictorial data recognition.

4.5.1 TEDACluster for Big Data

The industrial applications provide larger demand for the big data processing as the computing system get more and more computational powers. It gives a reason for the development of the clustering and classification techniques for Big Data applications. The algorithms for BigData should be capable of data streams processing for the initially unknown size, as well as providing scalability in order to effectively utilise existing hardware facilities.

This section explains the adaptation of the TEDACluster technique, proposed in section 4.1.4, for the BigData applications.

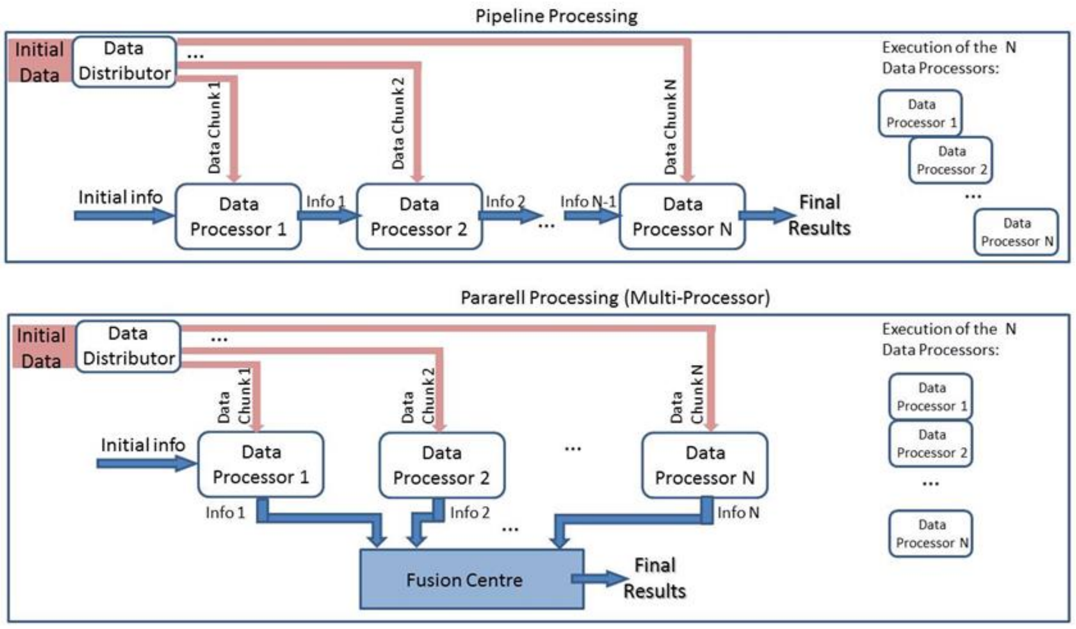


Figure 21 Data processing pipeline [21]

Two architectures of BigData processing techniques were introduced (figure 21) in the paper [21]:

- Pipeline processing considers that there are different data flows, which do not interfere with each other. Each node receives its own chunk of data, and processes it only when its turn comes in the pipeline. The time of idle for one pipeline may be utilised for another one. As the order does not change in this case, this approach produces exactly the same results are for the sequential processing on one node.
- Parallel processing considers the situation when the data is distributed between several nodes and are processed there independently. In this case, the same order cannot be provided (it can be not so critical for practical applications), however such approach avoids idle stay of the nodes.

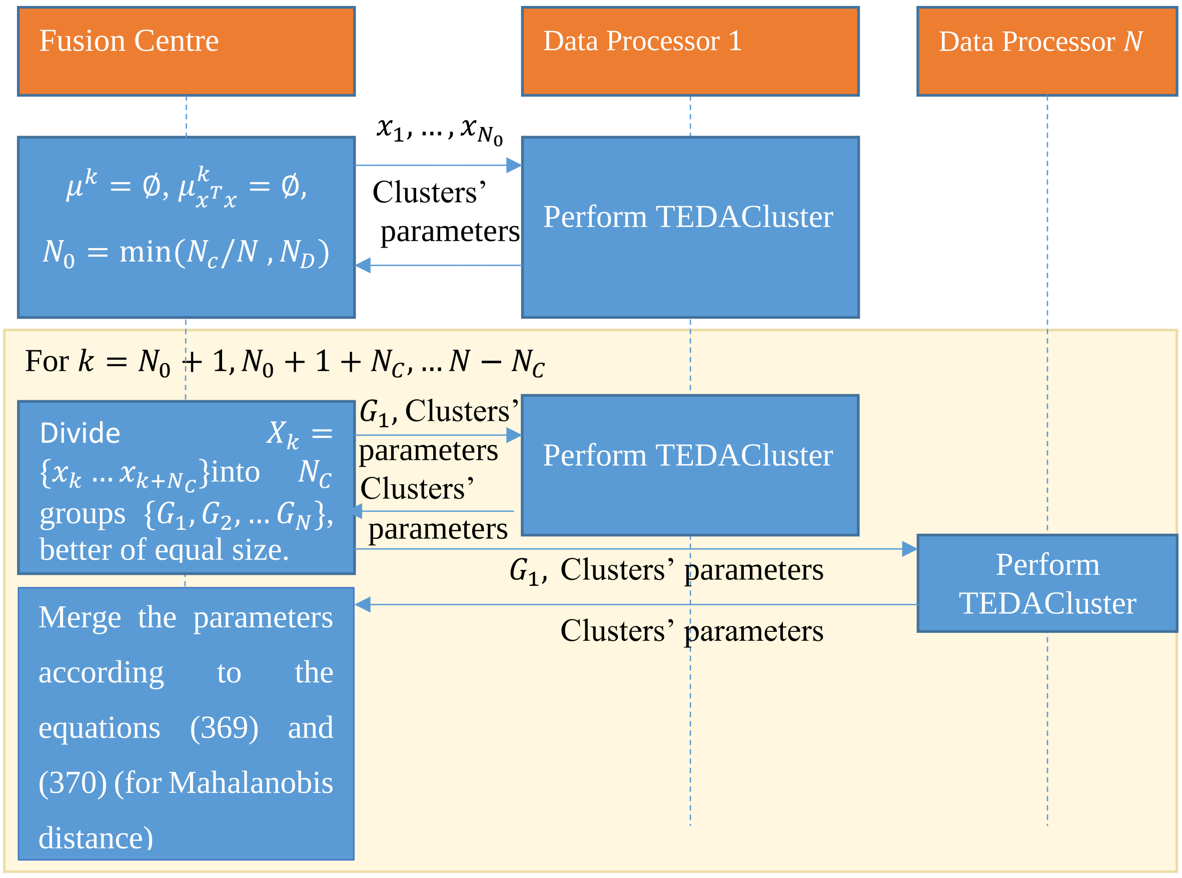

Fusion Centre
Data Processor 1
Data Processor N
$x_1, \dots, x_{N_0}$
$\mu^k = \emptyset, \mu^k_{x^T x} = \emptyset,$
$N_0 = \min(N_c/N, N_D)$
Clusters' parameters
Perform TEDACluster
For $k = N_0 + 1, N_0 + 1 + N_C, \dots N - N_C$
Divide $X_k = \{x_k \dots x_{k+N_C}\}$into N_C groups $\{G_1, G_2, \dots G_N\}$, better of equal size.
G_1, Clusters' parameters
Clusters' parameters
Perform TEDACluster
G_1, Clusters' parameters
Perform TEDACluster
Clusters' parameters
Merge the parameters according to the equations (369) and (370) (for Mahalanobis distance)

Figure 22 TEDACluster for Big Data

The scheme, depicted in Figure 22, explains the proposed TEDACluster computation scheme for Big Data. In this algorithm the data is divided into chunks by the Fusion Centre. Here, N is a data processors number, X is a data set, N_D is the number of data set elements, N_c is the data chunk size. The first one is processed on the first node to initialise the model, and all the subsequent chunks are divided between the nodes. After responses from all the nodes are received by the Fusion Centre they are being merged into the model. All the clusters from all the models are added to the overall model, and then the close clusters from the set R_d are merged:

$$R_d = \{R_i : j > i, R_j \in F, t_j^k(\boldsymbol{\mu}_i^k) > T(k)\}, \tag{368}$$

The merging process in the Fusion Centre is given by the following formulae if considering Mahalanobis distribution:

$$\boldsymbol{\mu}_l^0 = 0, \boldsymbol{\mu}_l^k = \frac{k - N_{i,j}}{k} \boldsymbol{\mu}_l^{k-N_{i,j}} + \frac{N_{i,j}}{k} \boldsymbol{\mu}_{i,j}^{N_{i,j}}, \tag{369}$$

$$\boldsymbol{\mu}_{(xx^T)_l}^0 = 0, \boldsymbol{\mu}_{(xx^T)_l}^k = \frac{k - N_{i,j}}{k} \boldsymbol{\mu}_{(xx^T)_{i'},j}^{k-N_{i,j}} + \frac{N_{i,j}}{k} \boldsymbol{\mu}_{(xx^T)_{i'},j'}^{N_{i,j}} \tag{370}$$

Here $\boldsymbol{\mu}_{i,j}^{N_{i,j}}$ is the mean of the i-th cluster in the j-th processor, l is the index of the cluster from the Fusion Centre. The fuzzily reweighted least squares algorithm is formulated both for standard TEDACluster [19] and for previously proposed eClustering based models like those described in [17], [111], [115]. The experiments, carried out with this method (as a part of TEDAClass implementation), are described in section 5.3.2.

4.5.2 TEDAClass for Big Data

Based on the TEDACluster algorithm for Big Data described in section 4.5.1, the Big Data version of TEDAClass is presented here [21]. The difference is that additionally to clustering the data is labelled after being clustered. This labelling is done using the linear regression model for each of the clusters using fuzzily reweighted recursive least squares method, described in section 4.2.1. The algorithm uses the same scheme of delegation of calculations from the Fusion Centre to the Data Processors as it was described in section 4.5.1, but executes TEDAClass algorithm, described in section 4.2.1, on each of the nodes instead of TEDACluster. The experiments for this method are described in section 5.3.2.

4.6 Conclusion

In this chapter the proposed object detection and classification methods have been described. First, the recently proposed TEDA approach [159] has been reviewed in section 4.1.1, giving a basis to TEDA family of classifiers, proposed and described in this chapter. For this approach, the recursive computation schemes for the TEDA framework quantities have been proposed in sections 4.1.2 and 4.1.3, particularly emphasising the necessity of recursive covariance matrix update avoiding matrix inversions.

Based on the general TEDA framework and fuzzy rules structure, discussed in section 2.2.5, the TEDACluster method was proposed. The TEDAClass and TEDAPredict methods are formulated in sections 4.2.1 and 4.2.2, based on the same idea, but allowing supervised training due to the usage of the recursive least squares method.

Then the exact recursive update scheme for the SVM method [98] has been proposed in order to address the problem of SVM training, when kernels and box constraints are derived from data. The motivation for training these components of the SVM problem is to take into account the abnormality quantities for each of the elements of the data set and, at the final stage, ensure that the majority of (the least abnormal) vectors are classified right at the first stage. The examples of a trainable kernel and box constraints, based on the TEDA quantities, have been given.

Another possible approach for object detection is to use image segmentation techniques. For this purpose, the improved technique, based on Chan-Vese functional, has been proposed in section 4.4 in two versions, parametric and non-parametric.

One of the spin-off topics, which has not been initially included into the research plan, but is extremely attractive for the contemporary video analytics, is Big Data. For this purpose, the architectures for the Big Data implementations of TEDA-based algorithms for clustering and classification have been elaborated in section 4.5.

The variety of methods, described in this chapter, is justified by the diversity of practical and theoretical problems of video analytics and the need for miscellaneous methods in order to cope with them. However, these methods are unified by the idea of real-time processing, and, for most of them, recursive computation, which is extremely useful when there is a need in a dynamic model update.

5 Implementation and validation of the developed algorithms

In this chapter the performance evaluation for the proposed algorithms is presented, as well as the practical algorithm application examples. The chapter is organised as follows. The video tracking algorithms, proposed in chapter 3, are assessed in section 5.1. The experiments with the clustering technique TEDACluster, described in section 4.1.4, are given in section 5.2. The results of the classification with TEDAClass (section 4.2.1) and TEDA SVM (section 4.3) methods, are given in sections 5.3.1 and 5.3.3. The experiments Big Data implementation of TEDA, described in section 4.5, are described in section 5.3.2. The experiments with TEDAPredict regression technique, proposed in section 4.2.2, has been described in section 5.4. The image segmentation algorithm, based on the modified Chan-Vese functional in its parametric and non-parametric version (section 4.4.1), are described in section 5.5. The chapter is finalised by the conclusion (section 5.6).

5.1 Tracking algorithms results and applications

The tracking algorithms, described in section 3, have been assessed in different ways. First, the assessment of the algorithm's capabilities to track and detect the objects on standard data sets for object tracking are described (section 5.1.1). Then the evaluation of the algorithm's ability of moving vehicles tracking is shown in section 5.1.2. Finally, incorporation of the object tracking algorithms into the marine object tracking system is reflected in section 5.1.3.

5.1.1 Moving objects detection and tracking assessment

The object detection and tracking algorithm, proposed in section 3.2, was assessed for vehicles (cars and lorries) tracking for the North-West Aerospace Alliance GAMMA programme [208]. The concept of such algorithm also repeats the one described in the patent which has been filed during this project [209] (see figure 23).

The following assumptions were taken into account in the tests in the project (technically, the assumptions for the algorithm, proposed in this thesis, are described in section 3, all other are assumptions, corresponding to the system implementation restrictions, as it was declared for the GAMMA programme):

– rich background (town, village, city, rural road)

– various types of surface where the objects of interest appear (tarmac, earth, sand road)

– only moving objects with sharp borders are in a scope

– the object size should have a size at least 100 pix^2

– the video should be captured in a clear day (no fog, mist, dust, rain, or snow) during daylight hours, with illumination level from 500 lux to 100 000 lux

– the system hardware shall support input video stabilisation

– to obtain better objects' geographical co-ordinates estimation, the camera should take a top-down position, with absolute value of the angle with the earth surface direction not exceeding 45 degrees.

– for a proper estimation of the objects' co-ordinates, the flat terrain is assumed (no mountains and hills)

– the assumed camera resolution is up to 720p with 10 to 30 frames per second (FPS)

– the telemetry sensor frequency should be at least double as FPS rate, and the telemetry and video data synchronisation has accuracy up to $\frac{1}{2 \times FPS}$ seconds.

– the object of interest's speed is not lower than 10 pix/sec.

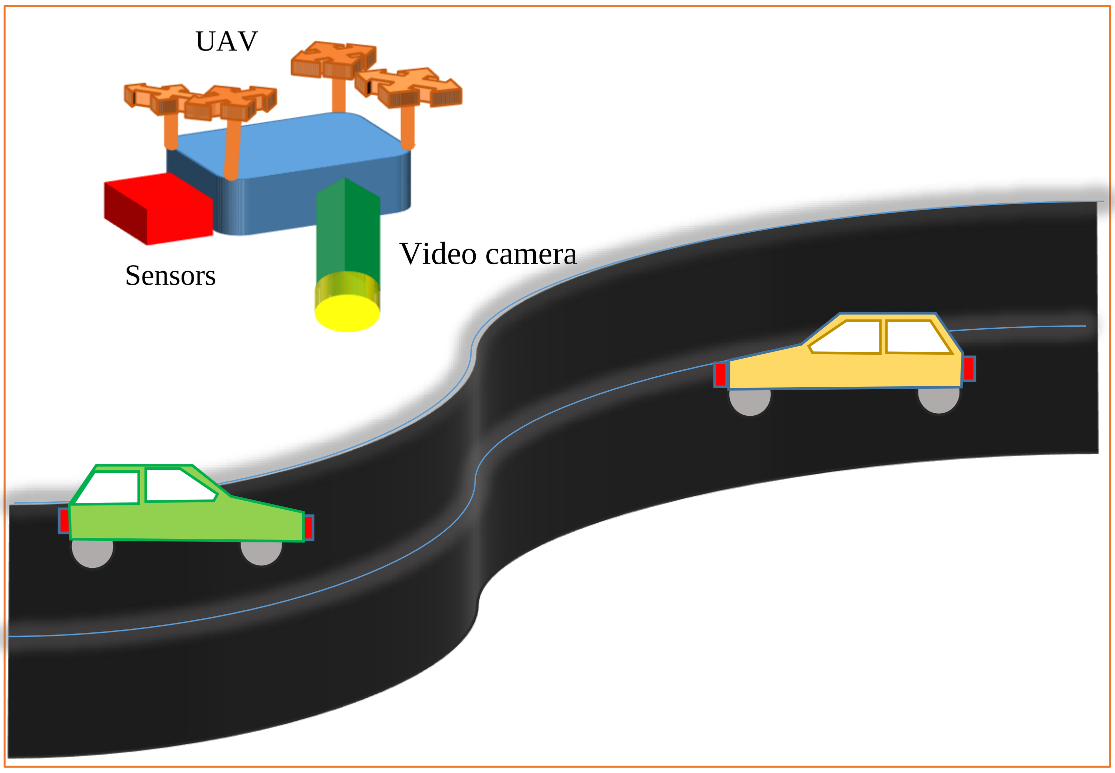


Figure 23. The output of the object detection and tracking algorithm.

The system installation for the benchmarking (see figure 24) included the following components:

1. the camera and the gimbal: the camera's area of view is approximately marked in the image;
2. the computer: accepts the data from the gimbal camera after digitation provided by the Video Encoder
3. the INS Sensor (was not integrated and not used in the experiments, however it was partially supported by the software)

4. the GPS Receiver for the INS (was not integrated and not used in the experiments)
5. the AXIS Q7401 H.264 Video Encoder
6. the LCD Screen - provides a platform for the system control
7. the mouse --- provides control over the bench testing system
8. The keyboard --- provides control over the bench testing system.

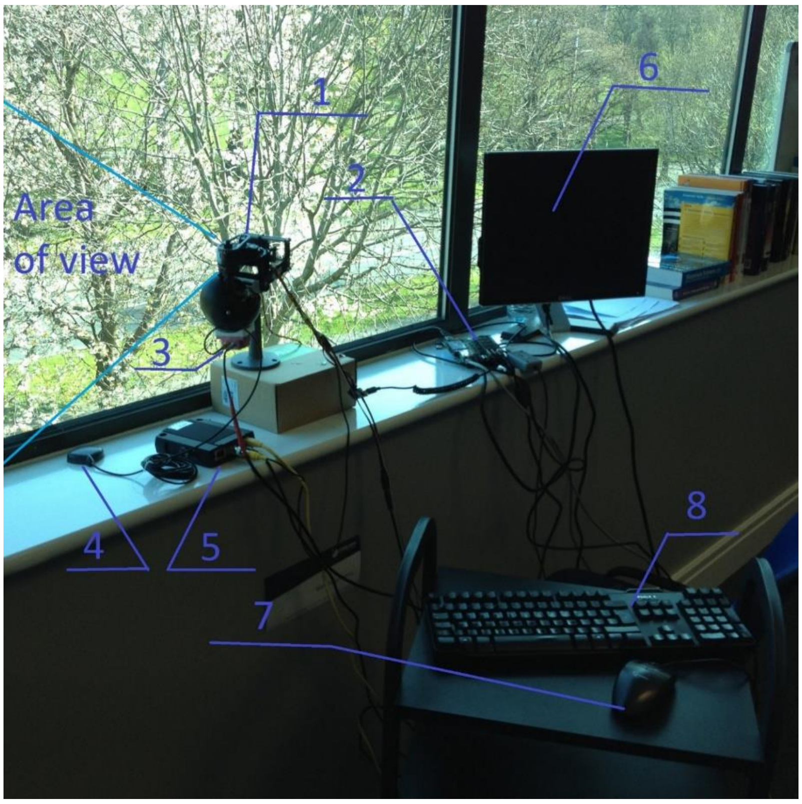


Figure 24. The system installation for bench testing.

The following data set were used for the performance metrics tests:

- BOBOT dataset [210], [211], video files Vid_A_ball.avi (figure 27), Vid_B_cup.avi (figure 28);
- UCF Aerial Action Data Set [212], files sequence_1.avi (figure 25), sequence_2.avi (figure 26).

For both the data sets, several performance indicators were assessed using ground truth data which was provided within the data sets. BOBOT dataset has been designed for single object tracking. The UCF Aerial Action Data Set was designed for multiple object tracking with reappearance.

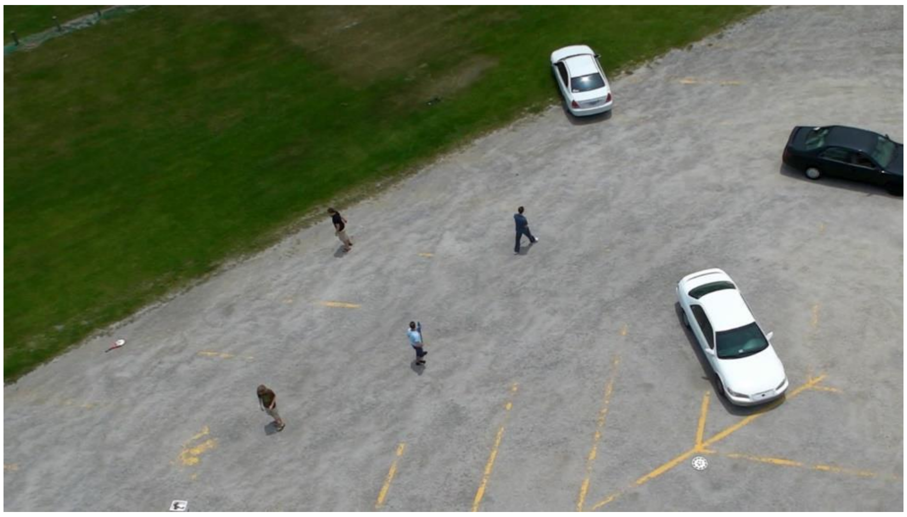


Figure 25. Video frame from the file Sequence_1.avi from UCF Aerial Action Data Set [212]

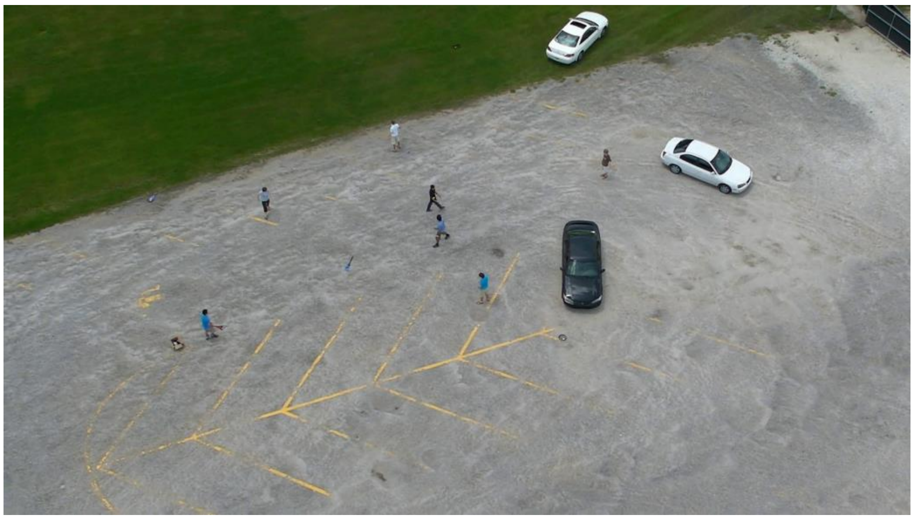


Figure 26. Video frame from the file Sequence_2.avi from UCF Aerial Action Data Set [212]

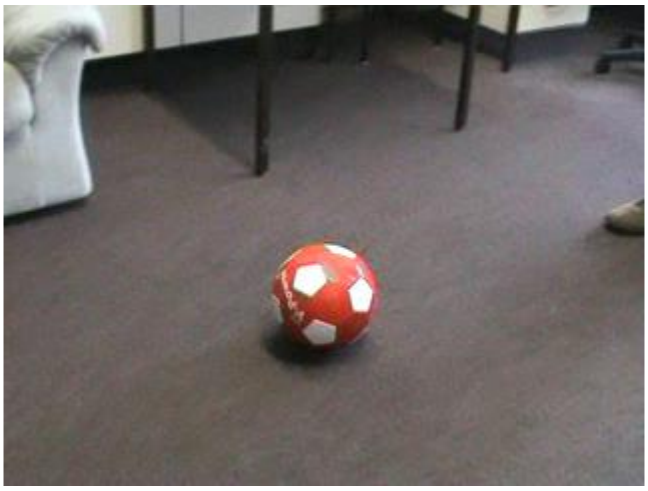


Figure 27. Video frame from the file Vid_A_ball.avi from BOBOT dataset [210], [211]

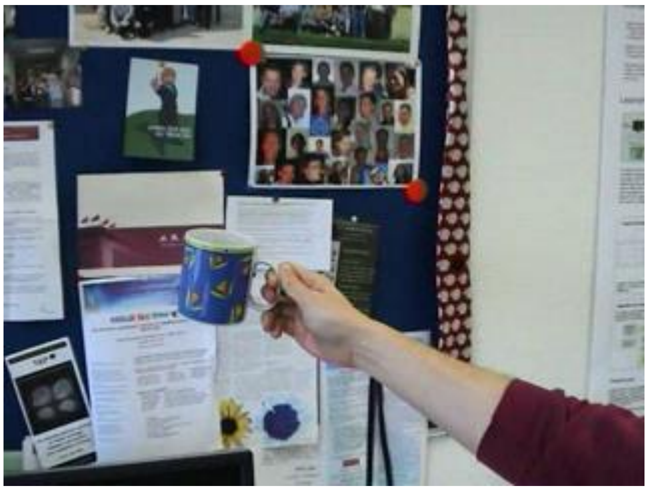


Figure 28. Video frame from the file Vid_B_cup.avi from BOBOT dataset [210], [211]

In order to evaluate the performance of the system in the practical scenario, the set of scores has been developed in order to assess the success of the algorithms' application. The scores were selected according to the following criteria:

- the score should be quantitative when it is possible, i.e. reflect the results of the measurements in numbers, and
- the success criterion for each of the scores is to be defined by a range of values.

The scores used for the assessment are present in table 1, which was developed for the GAMMA programme [208].

Table 1 Scores for the tracking algorithm quality assessment on the standard data sets

Id	Name	Description	Measurement Unit	Success Criterion
Detection				
1.1	Detection:: Effectiveness	Moving objects fraction over total	Double [0, 1]	> 0.8
1.2	Detection:: EffectivenessPerFrame	Fraction of frames containing objects' appearance, where the objects were detected (for single object tracking)	Double [0, 1]	> 0.8
1.3	Detection:: FalseAlarmRate	False positive detection rate: #[False objects detected] / #[Objects detected] (for multiple object tracking)	Double [0, 1]	< 0.15

Miscellaneous				
2.1	Misc:: FrameProcessingTime	Data processing time	milliseconds (double)	100 ms when video is up to 720p lines
2.2	Misc::Latency	Delay between the frame appearance and its return after the processing	milliseconds (double)	Latency ≤ 150 ms

In table 2 these metrics were assessed on the video file Vid_A_ball.avi (figure 27) from the BOBOT data set [210], [211].

Table 2 Scores of the tracking algorithm quality assessment on the Vid_A_ball.avi file

Run	Name	Threshold	Result	Comment
S1-1.1	Detection::Effectiveness	> 0.8	1	Success
S1-1.2	Detection:EffectivenessPerFrame	> 0.8	0.8142	Success
S1-2.1	Misc::FrameProcessingTime	< 100 ms when video is up to 720p lines	49 ms	Success
S1-2.2	Misc::Latency	Latency ≤ 150 ms	49 ms	Success

The object detection flag graph for the video file Vid_A_ball.avi is shown in figure 29. This flag shows whether the object was detected for the frames where the object was actually present (1 if the object has been detected, 0 otherwise). The parts of the graph, when the object is not being detected, correspond to the initial period of object localisation after the object reappearance.

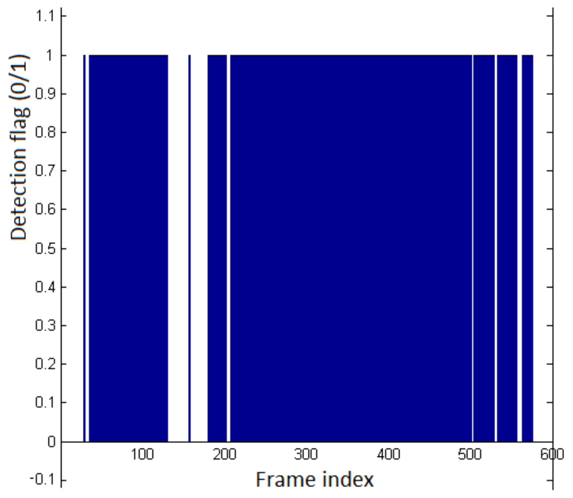


Figure 29. The object detection flag for the video file Vid_A_ball.avi

For the file Vid_B_cup.avi the same test was carried out. The test results are present in table 3.

Table 3 Scores of the tracking algorithm quality assessment on the Vid_B_cup.avi file

Run	Name	Threshold	Result	Comment
S2-1.1	Detection::Effectiveness	> 0.8	1	Success
S2-1.2	Detection:EffectivenessPerFrame	> 0.8	0.8624	Success
S2-2.1	Misc::FrameProcessingTime	< 100 ms when video is up to 720p lines	46 ms	Success
S2-2.2	Misc::Latency	Latency ≤ 150 ms	46 ms	Success

The object detection flag graph for this file is shown in figure 30.

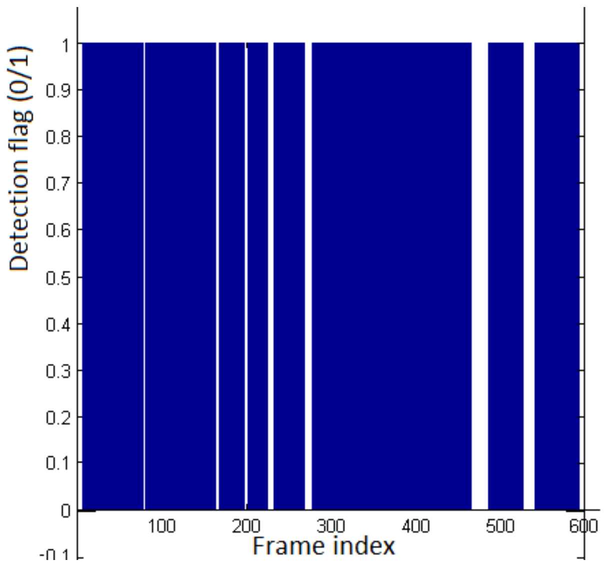


Figure 30. The object detection flag for the video file Vid_B_cup.avi

The results of the experiments for the files Sequence1.avi and Sequence2.avi are shown in tables 4 and 5.

Table 4 Scores of the tracking algorithm quality assessment on the Sequence1.avi file

Run	Name	Threshold	Result	Comment
S3-1.1	Detection::Effectiveness	>0.8	1	Success
S3-2	Detection:: FalseAlarmRate	<0.15	105 /740≈ 0.1419	Success
S3-6	Misc::FrameProcessingTime	<100 ms when video is up to 720p lines	47 ms	Success; the video file was scaled
S3-7	Misc::Latency	Latency ≤ 150 ms	47 ms	Success; the video file was scaled

Table 5 Scores of the tracking algorithm quality assessment on the Sequence2.avi file

Run	Name	Threshold	Result	Comment
S3-1	Detection::Effectiveness	>0.8	1	Success
S3-2	Detection:: FalseAlarmRate	<0.15	28 / 269≈ 0.1041	Success
S3-6	Misc::FrameProcessingTime	<100 ms when video is up to 720p lines	49 ms	Success; the frames were resized
S3-7	Misc::Latency	Latency ≤ 150 ms	49 ms	Success; the frames were resized

5.1.2 Moving vehicles tracking and detection

The moving object tracking framework has been assessed in different conditions (see [15]); the information about these conditions is provided below. First, some of the tests were performed with benchmark data. One of the benchmarks is VIVID PETS 2005 data set [167] (see figure 31) which contains multiple moving vehicles captured from the moving airborne vehicle detection camera. For one particular object, the video is supplied with the ground truth data for one object for every tenth frame.

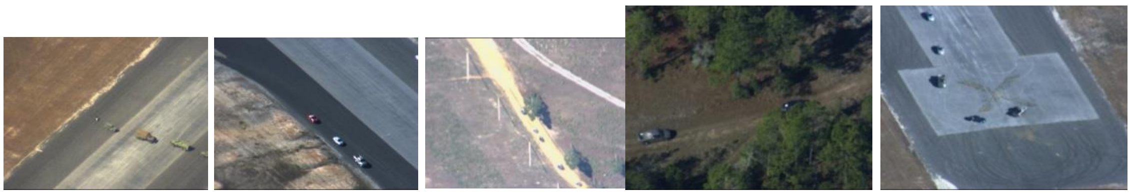


Figure 31. VIVID PETS 2005 data set sample frames

To make a comparison with a previously existing method [168], the experiment from this article is reproduced. In this experiment, the 'Match' metric is defined as a fraction of the frames where the object's bounding box contains the ground truth data (the closer to 1 the better). The 'Size ratio' metric is an average ration between the area of the detected bounding box and the ground truth one (the closer to 1 the better). The final results, obtained with the maximal quantity of the object clusters 30 and the object detection parameter 10, are described in the table 6 which reproduces the results described in the article [168] and given for the algorithm described in sections 3.2.4 and 3.3.

Table 6 The comparison of the proposed algorithm with the method [168]

	Match (method from section 3.2.4)	Size Ratio (method from section 3.2.4)	Match (method from section 3.3)	Size Ratio (method from section 3.3)	Match (Method and data from [168])	Size Ratio (Method and data from [168])
EgTest01	**0.9828**	2.57	0.9717	2.59	0.9500	1.00
EgTest02	**0.9302**	2.47	0.9225	3.13	**0.9302**	1.23
EgTest03	**0.9337**	2.06	0.8466	1.12	0.8588	0.78
EgTest04	**0.9302**	3.51	0.9005	3.63	0.6000	1.19
EgTest05	**0.9080**	0.49	0.8642	0.54	0.8889	0.88

The results (sample is shown in Figure 32) show the stable behaviour of the algorithm in different scenarios (one can see that the detection rate is higher than 90% for the method from section 3.2.4). The size ration is higher because the optical flow is higher in the vicinity of the objects and the cluster needs to adapt to the object on several frames and even insignificant change of the bounding box size can substantially impact the size ratio.

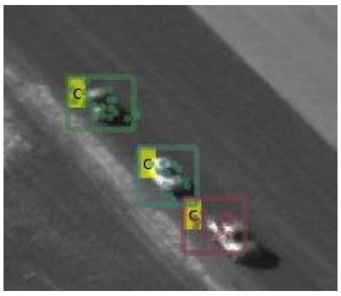


Figure 32. The output of the object detection and tracking algorithm.

5.1.3 Marine objects tracking

The system was successfully tested for the marine objects detection and tracking, described in the patent [209], filed during this research. The system concept depicted in Figure 33, and the detailed system structure is shown in Figure 34.

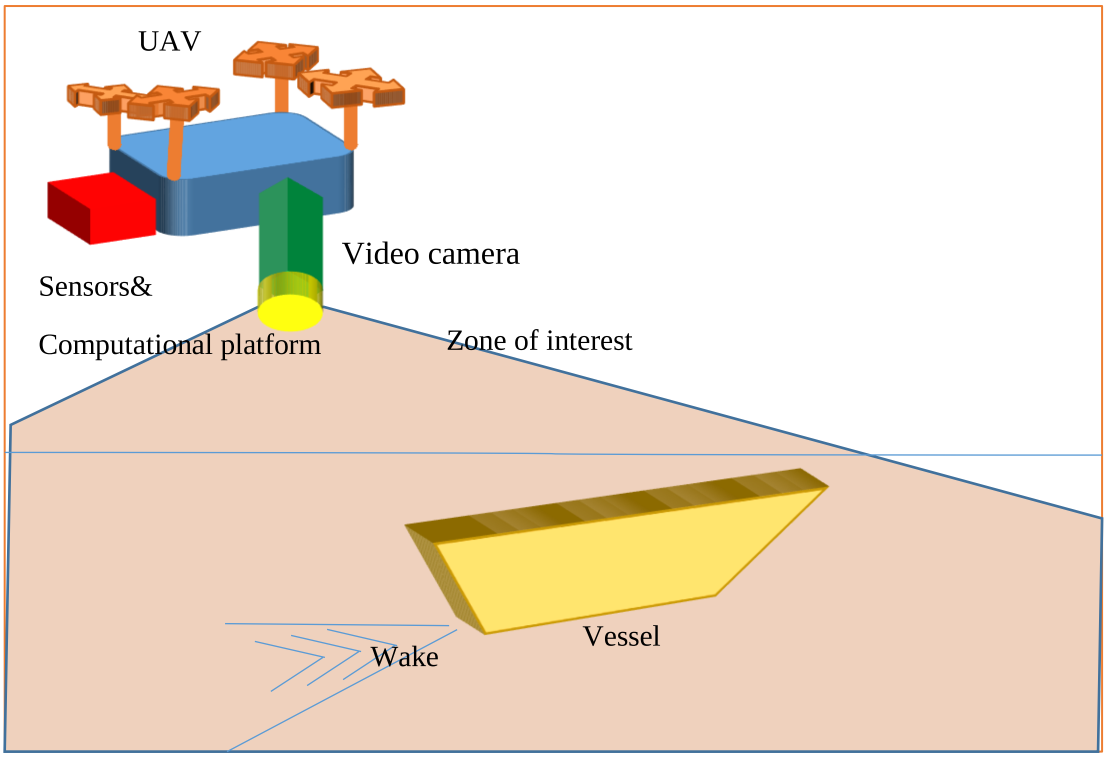


Figure 33 The concept of the marine object detection and tracking system

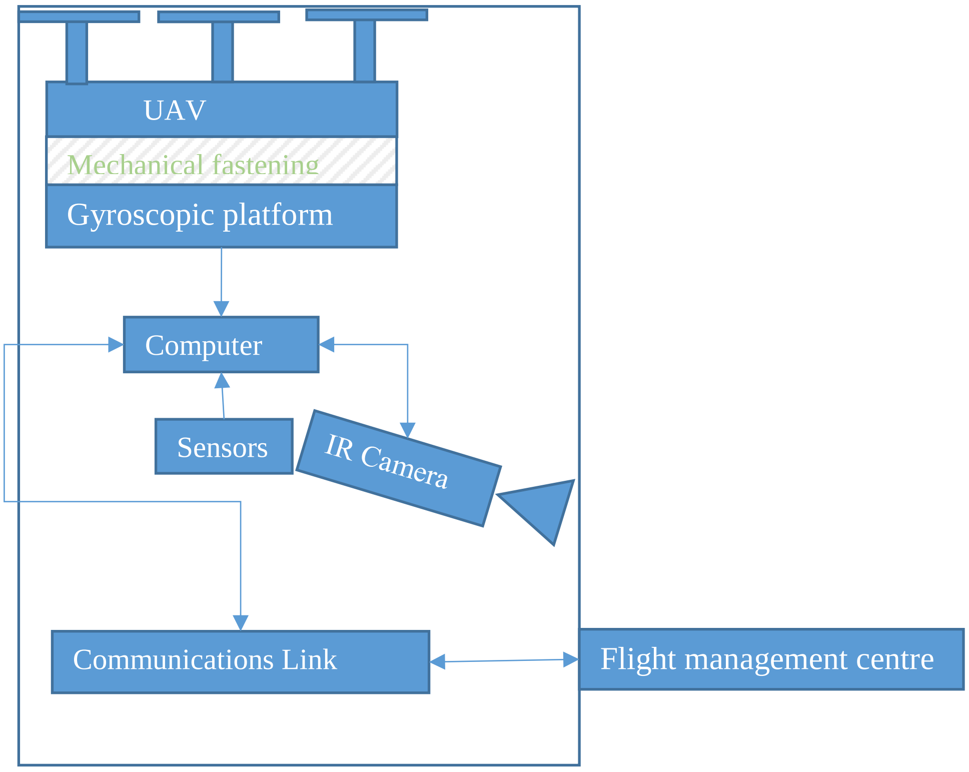


Figure 34 The component scheme of the marine object detection and tracking system

As it can be seen from Figure 34, the system consists of the following components:

- a UAV;
- a gyroscopic platform;

- a sensor set, which can include an accelerometer, inclinometer, GPS and, possibly, range finder for distance estimation quality enhancement;
- an infrared camera.

A UAV is used to bring all the sensors and the computational platforms to the required installation position. The connection to the UAV from the ground is to be provided by the wireless network, which is used for the telemetry (that is sensor measurements and the algorithms output) and video transmission. If the video transmission is impossible due to low bandwidth, then the telemetry may be transmitted.

Additionally, to measure distances to objects according to the approach given in section 3.6, the sensors are used to provide Euler angles and elevation. These sensors can be used to determine real world co-ordinates of an object. The overall object detection and co-ordinate estimation algorithm can be used to influence a flight plan, forcing the UAV to follow the object.

As it is seen from chapter 3, the algorithms are designed for unsupervised object detection and tracking, rely on a single camera video (in this application, it is thermal camera) and do not require additional cameras. The programmatic modules for these algorithms can be installed to a computational platform on a moving airborne vehicle (possibly UAV).

The following practical problem statement restrictions, which are addressed by the algorithms, described in chapter 3, have been imposed on the objects being detected:

- objects' borders must be sharp in order to correctly discern objects from background;
- a discernible object movement;
- rigidity of an object's movement: an object is shown as a continuous area that cannot change its form rapidly (but it can be accepted some slow form change due to co-ordinate projection from the three-dimensional world to the two-dimensional camera co-ordinate space.

5.2 Experiments with the clustering algorithm TEDACluster

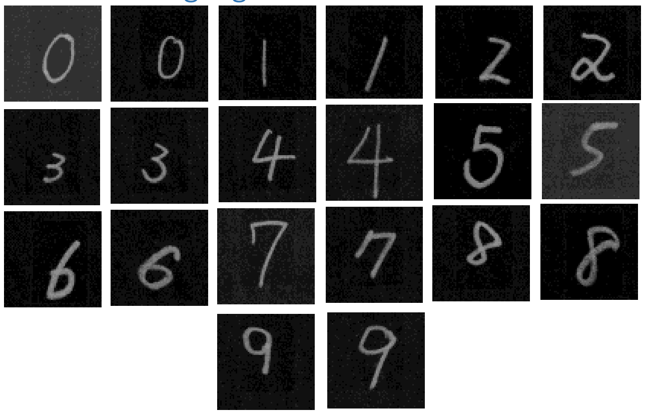


Figure 35. Symbol data samples (ETL1 database [175])

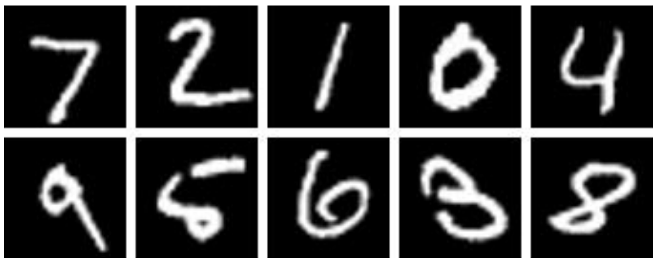


Figure 36 Symbol data samples (MNIST [176])

One of the widely known ways to assess the classifier performance is the symbol classification problem. For this purpose, two different well known handwritten symbol datasets: ETL1 [175] (Figure 35) and MNIST [176] (Figure 36). First, in order to assess the clustering quality of the clustering algorithm TEDACluster [19], the cluster purity Π for these datasets is calculated using formula:

$$\Pi = \frac{1}{k}\sum_{i\in K}\max_{j\in C}\left|\Xi_{ij}\right| : \Xi_{ij} = \left(y_p = j \,\&F\left(x_p\right) = i\right), \tag{371}$$

where $1 \leq p \leq k, x_p \in \mathfrak{X}$ is a particular data sample, y_p is a class identifier, corresponding to the sample x_p, $\mathfrak{X}$ is a data sample domain set, k is the data set cardinality, $F\left(x_p\right): \mathfrak{X} \to K$ is a fuzzy rule system as described in section 2.2.5, which can be represented as a function between the data samples space $\mathfrak{X}$ and the finite number of cluster labels K, and the data samples x_p are mapped into the ground truth class labels y_p. The clustering results shown in the table 7 repeat those published in the article [19]. One can see that the clustering purity, as it is expected, increases with the training set size increase.

Table 7. Clustering results for ETL1 data set [175]

Training set size	*Purity* Π	*Clusters number*
50	0.4400	15
100	0.5500	29
150	0.8000	82
200	0.6400	52
300	0.7833	101
400	0.7775	121
500	0.6700	77
600	0.7200	87
700	0.7143	97
800	0.7675	132
900	0.7689	128
1000	0.7750	179
1857	0.7878	201

5.3 Object classification experiments

5.3.1 Data classifier TEDAClass

Data classifier TEDAClass, which is described in section 4.2.1 and proposed in [19], was assessed on the same data sets, namely ETL1 [175] and MNIST [176], as TEDACluster, but for the classification problem. The results taken from the article [19] are given in tables 8 and 9. On the first test case (table 8), the method has been compared against the neocognitron neural network [77] which was specially fitted to the dataset. The method shows good results comparing to the rival algorithms and outperform all of them but neocognitron. For the second test scenario, on another dataset, the method outperforms all the methods including neocognitron, that shows good generalisation capabilities of the proposed method and applicability to wide range of practical problems.

Table 8. Recognition results for ETL1 data set [175]

Training set size	***eClass1***	***AutoClass1***	**Neocognitron**	***TEDAClass***
200	70.02%	86.67%	90.60%	89.09%
300	57.83%	89.80%	93.51%	91.36%
400	59.04%	91.71%	93.75%	92.93%
500	59.96%	91.59%	93.44%	92.91%
600	62.93%	92.39%	94.95%	93.27%
700	83.39%	92.92%	94.61%	93.74%
800	83.29%	93.05%	95.59%	94.06%
900	84.33%	93.06%	95.53%	93.95%
1000	75.20%	93.14%	95.15%	94.25%
1857	92.19%	95.23%	96.43%	96.08%

Table 9 Recognition results comparison for MNIST[176] database

Training set size	***eClass1***	***AutoClass1***	**Neocognitron**	***TEDAClass***
500	93.26%	94.53%	92.36%	**95.18%**
1000	86.54%	95.82%	94.42%	**95.92%**
2000	96.42%	96.44%	96.04%	**96.70%**
3000	96.55%	96.50%	96.34%	**96.67%**
4000	96.62%	96.68%	96.62%	**96.88%**
5000	96.85%	96.91%	96.94%	**97.16%**
10000	97.19%	97.24%	–	**97.38%**
20000	97.32%	97.38%	–	**97.53%**
30000	97.46%	97.44%	–	**97.68%**
40000	97.45%	97.42%	–	**97.66%**
50000	97.46%	97.38%	–	**97.65%**
60000	97.46%	97.42%	–	**97.63%**

5.3.2 TEDAClass-BDp

Here the evaluation procedure is described for TEDAClass-BDp algorithm [21], which is a modification of TEDAClass algorithm for big data described in section 4.5.2. The data set consists of 12,888 handwritten digits images each labelled, 1857 of which are used for training and 11,037 for validation, and the algorithm is implemented in Matlab®.

Table 10 The accuracy and computational time for TEDAClass-BDp [21]

#Data Processors	Accuracy	#Data Clouds	Computational time compared to one node
1	0.9540	79	1
2	0.9517	103	0.3835
3	0.9491	227	0.2504
4	0.9490	232	0.2371
5	0.9428	454	0.1256

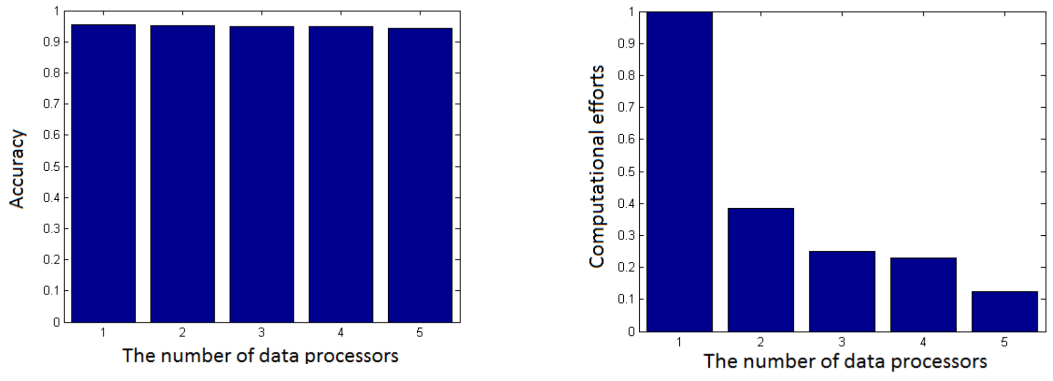


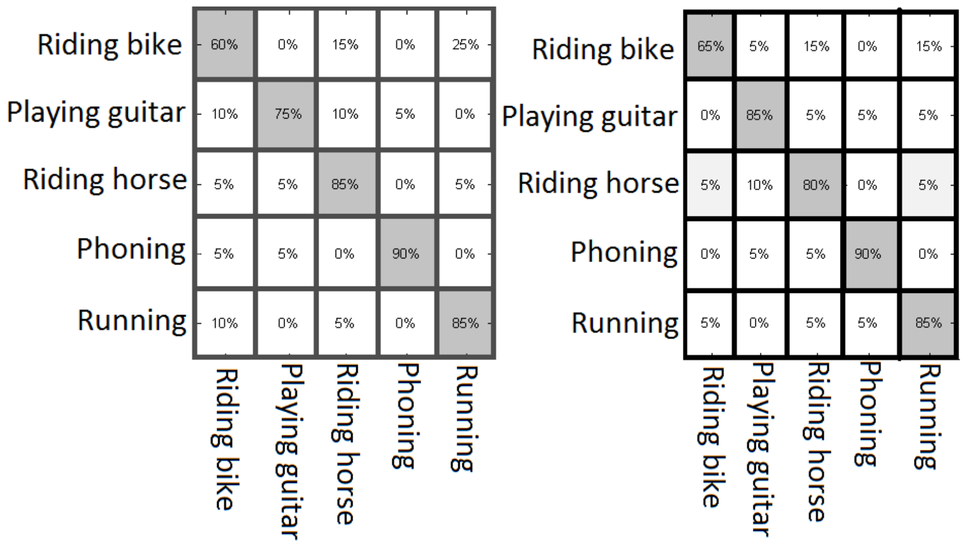
Figure 37 Accuracy (left) and Computational Time (right) for various numbers of data processors.

The feature extraction procedure is based on gist descriptor [182], exploiting Gabor features, with added Haar-like features [94] (see article [17]). The data is being partitioned between up to five Data Processors, and the time of processing is up to five times lower than for the sequential processing (that conforms with the theoretical estimations of $O(N)$ complexity and time reduction). To exclude the growth of the number of data clouds, the cluster merging is enabled (see section 4.5.2). The results of the experiments, depicted in Table 10 and Figure 37, are reproduced from the article [21].

The update time, as it can be seen from the figure 22 in section 4.5.1, depends linearly of the cluster (and that means, fuzzy rules) number. During the training stage, the number of clusters is increased because of the variety of the data patterns to be learnt by the classifier. The accumulation of the clusters is carried out of the Data Fusion Centre, which collects the clusters from the Data Processors and then merges them into the same model.

5.3.3 Human activity classification using SVM and TEDA

Figure 38. Human activities data set [199].



	Riding bike	Playing guitar	Riding horse	Phoning	Running
Riding bike	60%	0%	15%	0%	25%
Playing guitar	10%	75%	10%	5%	0%
Riding horse	5%	5%	85%	0%	5%
Phoning	5%	5%	0%	90%	0%
Running	10%	0%	5%	0%	85%

	Riding bike	Playing guitar	Riding horse	Phoning	Running
Riding bike	65%	5%	15%	0%	15%
Playing guitar	0%	85%	5%	5%	5%
Riding horse	5%	10%	80%	0%	5%
Phoning	0%	5%	5%	90%	0%
Running	5%	0%	5%	5%	85%

Figure 39. Results of the recognition for different methods (left picture is SVM with histogram intersection kernel, accuracy rate is 79%, right picture is SVM with TEDA kernel, combined with histogram intersection kernel, and TEDA box constraints, accuracy rate is 81%).

The TEDA SVM method, described in section 4.3 and proposed in [18], was assessed on the human activities data set [199] (Figures 38 and 39). The feature transformation is performed as it is described in article [17], consisting of Haar [94] and gist [182] features.

The following methods were compared in [18]:

- SVM with Gaussian kernel with $\sigma = 34.4725,\ C = 30$.
- SVM with TEDA kernel, combined with Gaussian kernel, with $\sigma = 34.4725, \gamma = 2$, $C = 30$, and TEDA weights.

The Gaussian kernel is given by the equation

$$K_G\ (x, y) = \exp(-(\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y})/(2\sigma^2)). \tag{372}$$

Contrary to the Gaussian probability density function, the kernel is not normalised because the solution does not depend of the normalisation constant.

TEDA kernel, combined with Gaussian kernel, is expressed as

$$K_{\text{TEDA}}\ (x, y) = K_G\ (x, y) \times \left(\left(\|\boldsymbol{x}-\boldsymbol{\mu}^k\|^2 + {\sigma^k}^2\right)\left(\|\boldsymbol{y}-\boldsymbol{\mu}^k\|^2 + {\sigma^k}^2\right)\right)^{\gamma}. \tag{373}$$

The training set size was 200 samples, or 40 images per each of five classes, and the testing set contained 100 samples, or 20 images per each of five classes.

Figure 39 shows the results for the Gaussian kernel SVM on the left and SVM with TEDA kernel, combined with Gaussian kernel, on the right, for the Human Activities data set [199] (see Figure 38). One can see that the results were improved comparing to the Gaussian kernel.

5.4 TEDAPredict regression experiments

To assess the TEDAPredict algorithm [19], the well-known wine quality marking data set [177] for white Portuguese wine was used, which was also featured in the article [178]. This description reproduces the results from the article [19]. The results are compared with the alternative methods [178], [179] preserving the assessment procedure. The data set is divided randomly with 5-fold cross-validation [180]. The rival approaches described in [178] provide the regression by means of multilayer neural network (NN), Gaussian kernel SVM [181], as well as linear/multiple regression (MR). The quality metrics are defined as

$$\text{MAD} = \frac{1}{|X_T|}\sum_{x\in X_T}\left|\hat{f}(x) - f(x)\right|, \tag{374}$$

$$\text{A}_\alpha = \frac{1}{|X_T|}\sum_{x\in X_T}\left[\left|\hat{f}(x) - f(x)\right| \le \alpha\right]. \tag{375}$$

Here X_T is the testing data set, the function to estimate is denoted as $f(x)$, and the regression function is $\hat{f}(x)$; $[\cdot]$ denote a predicate, which is 1 if true and 0 if false; α is a tolerance threshold. The regression evaluation results are depicted in table 11.

Table 11 Regression results [19], [178] for wine dataset [177]

	MR	**NN**	**SVM**	***TEDAPredict***
MAD	0.59 ± 0.00	0.58 ± 0.00	0.45 ± 0.00	0.5702 ± 0.00
$\boldsymbol{A_{0.25}}$	25.6 ± 0.1	26.5 ± 0.3	50.2 ± 0.3	29.49 ± 0.3
$\boldsymbol{A_{0.50}}$	51.7 ± 0.1	52.6 ± 0.3	64.3 ± 0.4	53.64 ± 0.4
$\boldsymbol{A_{1.00}}$	84.3 ± 0.1	84.7 ± 0.1	86.8 ± 0.2	85.15 ± 0.4

The results are proven to be decent comparing to the competing algorithms although they do not overcome widely renowned classifier SVM. But the algorithm has an advantage of evolving computation, which makes it competitive for data streams where the data patterns are changing during time.

5.5 Image segmentation experiments

In this section, the modified Chan-Vese algorithm performance and applications are presented. The original Chan-Vese algorithm [123] is compared with some standard algorithms, with the proposed version of Chan-Vese functional featuring enhanced optimisation technique, as well as with the non-parametric version of the functional proposed in the article [22]. The benchmarking data consists of the pictures of blood cells, and the task is to select the individual blood cells from the picture. Another data set is well-renowned Caltech101 data set [169], [170] (Figure 40).

Figure 40 The benchmarking images from Caltech101 dataset (from top to bottom, different image groups: ‘Buddhas’, ‘accordions’, ‘planes’).

Also, some benchmarking images from [172] were used. The comparison was carried out against some algorithms, described in sections 2.2.6 and 2.2.7, namely MacQueen’s k-means [61], Chan-Vese algorithm [123], graph cut segmentation algorithm [171], [172], and [173] exploiting the graph cut algorithm by Kolmogorov, Boykov and Zabih [164],[171], applied also to miscellaneous domains apart of image segmentation, and Otsu binarisation algorithm [124] as a standard baseline algorithm for binary image segmentation. For all these algorithms, if they require initialisation, the same initial segmentation Ξ was selected. The algorithm output assessment is carried out using the purity metric calculated using the formula

$$P=\sum_{i=1}^{C}\frac{N_i^d}{CN_i}. \tag{376}$$

where C is the number of clusters, N_i is a number of the elements of the i-th cluster, N_i^d is a number of elements in the majority ground truth class. The maximal and the best purity is 1 showing that each cluster represent only one ground truth class. The average purity P_A for the test set of M images is an average over the purities for each image P_i where i is the index of the current image:

$$P_A=\sum_{i=1}^{M}\frac{P_i}{M}. \tag{377}$$

The average purity weighted by per-image pixels count $N(I_i)$ is defined as

$$P_B=\sum_{i=1}^{M}\frac{P_iN(I_i)}{\sum_{j=1}^{M}N(I_j)}. \tag{378}$$

The results of the benchmarking are depicted in tables 12, 13, 14, 15 and represent the results given in the article [22].

Table 12 Segmentation purity for Caltech101 data.

	Purity	*Otsu*	*k-means*	*Chan-Vese*	*Graph cut*	*Modified Chan-Vese*	*Novel non-parametric method*
Airplanes	P_A	0.7387	0.7543	0.7508	0.7546	0.7663	**0.7669**
	P_B	0.7339	0.7523	0.7401	0.7520	0.7661	**0.7676**
Accordion	P_A	0.7122	0.7771	0.7473	0.7928	**0.8279**	0.8276
	P_B	0.7115	0.7725	0.7466	0.7920	**0.8326**	0.8323
Buddhas	P_A	0.7463	0.7186	0.7514	0.6969	0.7959	**0.7977**
	P_B	0.7466	0.7185	0.7466	0.6991	0.7934	**0.7952**

Table 13 Visual comparison of the segmentation results

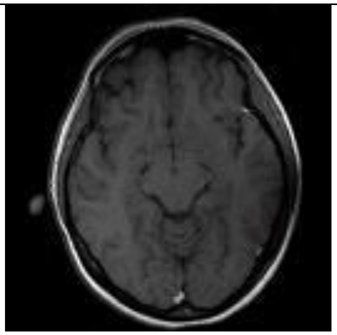
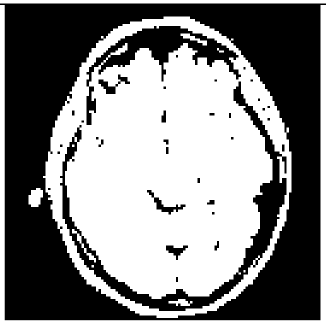
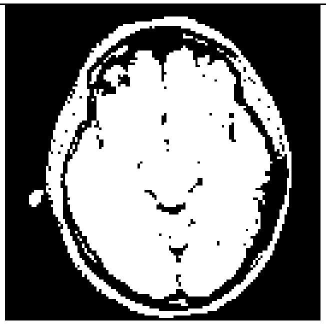
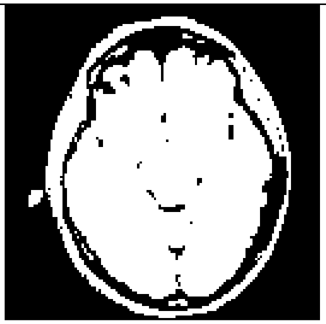
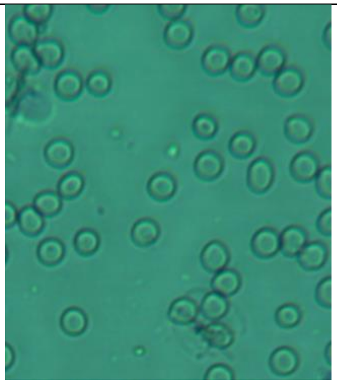
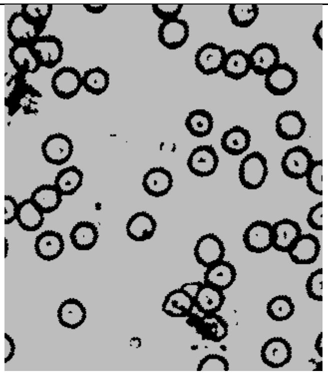
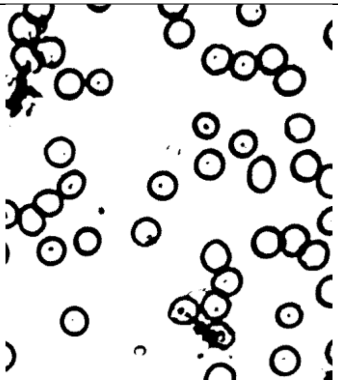
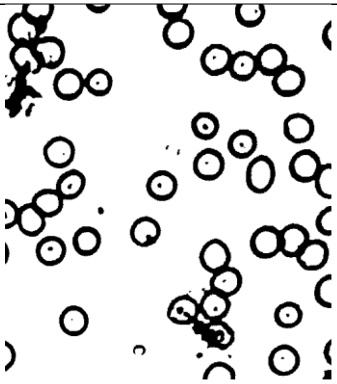
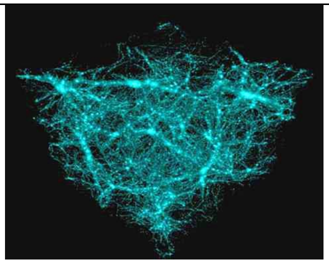
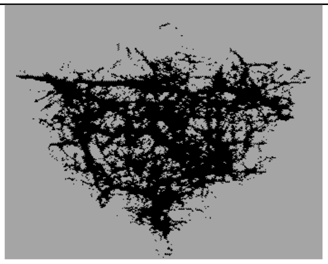
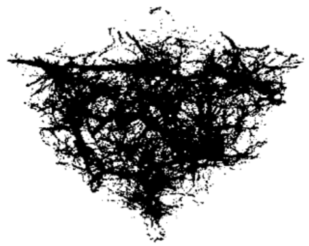
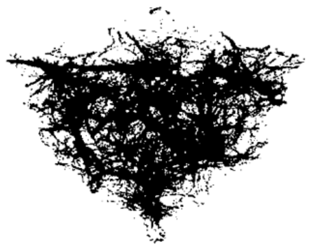
Original image	*Chan-Vese algorithm*	*Chan-Vese functional, graph cut optimisation*	*Non-parametric method*
			
			
			

Table 14 Algorithm convergence graphs

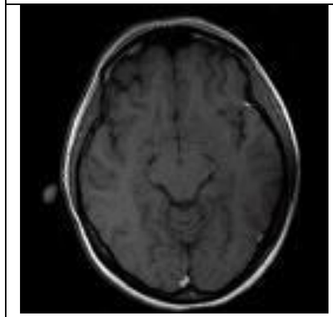
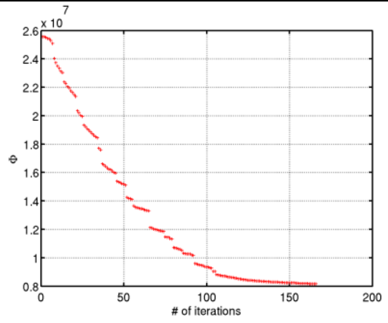
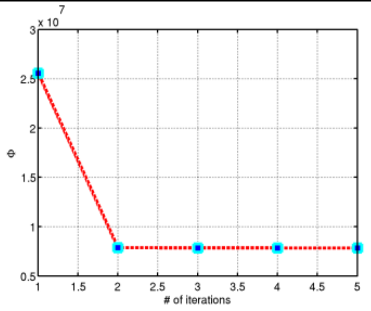
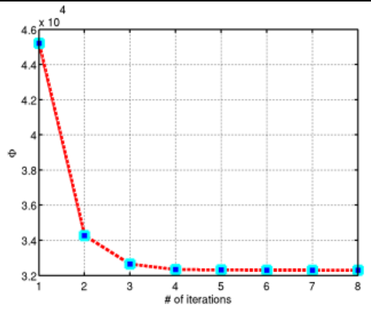
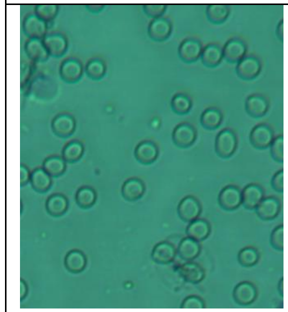
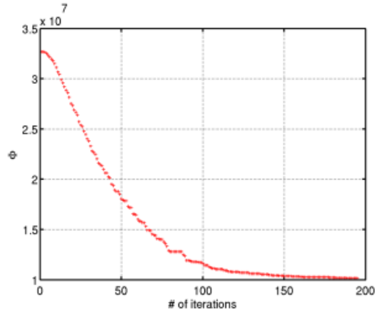
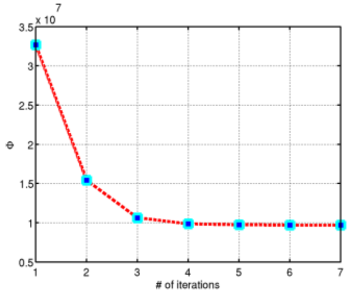
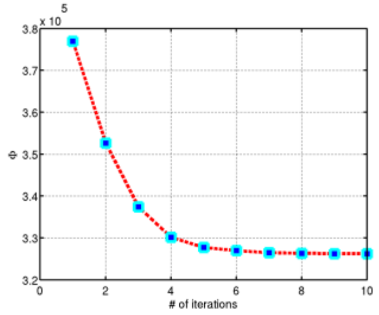
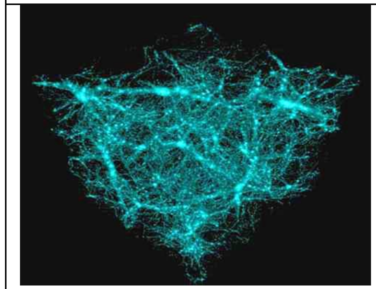
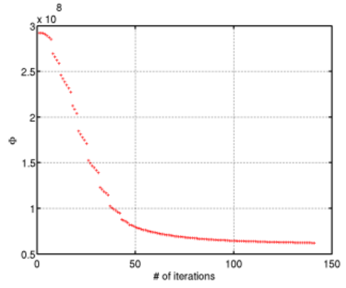
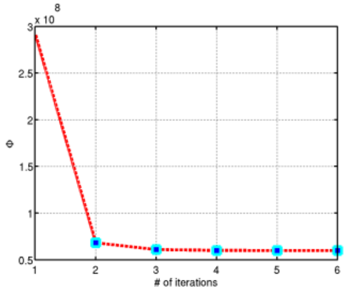
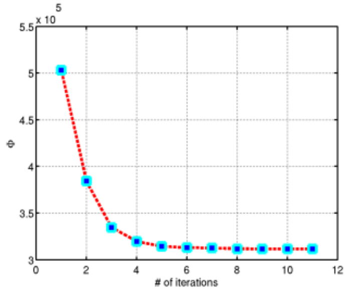
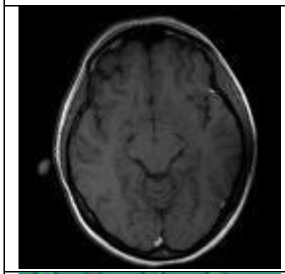
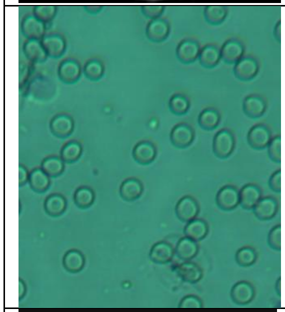
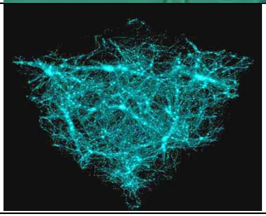
Original image	*Chan-Vese algorithm*	*Chan-Vese functional, graph cut optimisation*	*Non-parametric method*
			
			
			

Table 15 Convergence time and the number of iterations for different algorithms

Original image	*Chan-Vese algorithm*		*Chan-Vese functional, graph cut optimisation*		*Non-parametric method*
	Execution time, s	*The optimal functional value*	*Execution time, s*	*The optimal functional value*	*Execution time, s*
	1.25	8145095	0.13	7840105	0.1226
	19.85	10170074	2.29	9712299	1.8840
	8.37	62098824	1.20	59913013	0.9804

The results, depicted in table 12, show better purity for the proposed methods agains well-known ones. The visual results for some benchmarking images taken from [172] are present in table 13. The results differ because of the alternative minimisation procedure and exact Heaviside function instead of its approximation. The convergence graph in the table 14 shows significant reduction of the iterations number needed for the function optimisation: only several iteration are needed for the convergence instead of several hundreds. However, it should be emphasised that the functional values for the original and non-parametric functionals should not be compared because they correspond to different values therefore they are not present in the comparison table. Tables 12, 13, 14, 15 represent the results described in the article [22].

5.6 Conclusion

In this section, various practical use cases have been assessed, as well as tests with the proposed algorithms have been presented.

The experiments from section 5.1 have proven the applicability of the methods from section 3 to various practical and model scenarios in object tracking and moving object detection scenarios. The capability of the proposed methods to work in real time have been proven by the experiments. The comparison between two modifications of the method, with Laplacian and variational approximation of the update step, has been provided in section 5.1.2. All the experiments have been carried out with the video data and include a use case for thermal imaging.

The clustering algorithm TEDACluster, described in section 4.1.4, has been assessed on the symbol clustering problem in section 5.2. The classification algorithm TEDAClass, which has been proposed in section 4.2.1 and which is based on TEDACluster, has been assessed in section 5.3.1. The Big Data versions of the TEDACluster and TEDAClass have been assessed in section 5.3.2. The TEDA SVM algorithm has been assessed in section 5.3.3 on the human actions classification dataset against the state-of-the art SVM-based techniques. This study has shown the advantages of the proposed kernel and box constraints training against the traditional SVM-based methods.

Finally, the experiments on both parametric and non-parametric versions of the modified Chan-Vese functional, proposed in section 4.4, have been described in section 5.5, featuring comparison of the proposed methods with the state-of-the-art image segmentation techniques in terms of segmentation purity. The increased speed of the algorithms against the rival methods has been proven that shows the method’s capability to work in real time.

The revealed experiment results show robust performance of the proposed algorithms compared to the state-of-the-art algorithms and the perspective applicability of the proposed methods.

6 Conclusion and future work

This work is a synthesis of two topics, which were historically being developed predominantly independently, namely object tracking and pattern recognition. Original idea of considering multiple object tracking as a rigid motion segmentation problem evolved into the domain-independent Bayesian filter and a toolset of algorithms for video analytics. The object detection algorithms also join the ideas of evolving systems with fuzzy systems and TEDA frameworks.

6.1 Key Contributions

The results described in this thesis are manifold and united by the idea of the intelligent video surveillance:

- The novel multiple object tracking algorithm in two versions, based on Laplace and on variational approximation frameworks, was proposed. The approach is capable of multiple target tracking using time-consistent clustering.

  In contrast to most of the state-of-the-art algorithms, the proposed one does not distinguish between the object and clutter measurement on the tracking stage. The problem is solved after tracking stage by selecting those clusters that conform to the object model. The proposed multiple object tracking algorithm is defined in a domain-independent way and after then applied to the video analytics domain.
- The classification, clustering and regression techniques, based on promising data analysis framework TEDA, are proposed.

  The proposed techniques are based on TEDA framework and features fuzzy systems architecture and include versions for sequential and parallel data processing, taking into account the practical needs of big data processing. The methods also incorporate the abilities of evolving systems to adopt dynamically to changing statistical properties of the data that is particularly useful for data stream analytics.
- The incremental update procedure for C-SVM with changing kernels and box constraints is proposed, together with the learnable TEDA SVM kernel.

  The procedure enhances previously known incremental SVM algorithm, adding new features, such as incremental update with changing kernel and box constraints. The proposed TEDA SVM kernel is an example of the kernel that can be used with such incrementally trainable algorithm, but the approach is not restricted to the particular kernel type.

- The background subtraction technique based on velocity estimation is proposed within the Bayesian filter framework.

  The technique is used for detection the object of interest amongst the clusters, selected in a time consistent way using the proposed Bayesian filtering technique in two variants of implementation, featuring variational and Laplace approximations.
- The image segmentation algorithm based on Chan-Vese algorithm is proposed.

  The image segmentation technique enhances previously known active contours approaches by using MM algorithm approach, which is proven to be more effective in terms of the execution time and number of iterations. Besides that, the non-parametric version of the algorithm was proposed.
- The experiments for all the proposed tracking, clustering, classification and video surveillance techniques have been carried out.

  The results of the experiments on all the approaches named in the previous contributions are given in the experimental section.

6.2 Future work plans

The following approaches are to be considered in the future for the enhancement of the Bayesian filtering technique:

- The stereo vision can improve the accuracy of tracking and detection based on the velocity features, as well as improve the geographical co-ordinates estimation.

  The stereo vision approach, consisting of two cameras placed on rigid beam, relies on the feature point matching techniques to find out the correspondence between the points on different cameras, allows to accurately calculate distances to the objects without any assumption on the terrain form (the results are dependent only on the parameters of the cameras and the beam length). Used together with inertial navigation system, it can be used the geographical co-ordinates estimation, also it allows to calculate velocities of the objects.
- The filter can be adopted in the way which excludes using the parameter of the maximal number of clusters which is needed now.

  The proposed approach has a number of clusters parameter, which is not needed in many state-of-the-art applications. To factor out the parameter, the automatic relevance determination for EM algorithm (ARD EM [203]) could be used, but it is barely suitable for real-time applications, as it is computationally intensive. Another approach is to use the object detection algorithm before the tracking stage to determine the number of components of the Gaussian mixture, and then propagate it.

- The approach can be compared against well-known filtering techniques featuring JPDA [44], PHD [45] and MHT [43] data association.

  To make the comparison possible, there is a need to implement a detection algorithm. In the particular case of moving object detection, the visual odometry, or velocity measurement, for each of the feature points will be useful to factor out the clutter. Therefore, the JPDA-, PHD- or MHT-based model for fully automatic object detection and tracking can be also a separate contribution.

- The possibility of the combination of the proposed Bayesian filtering technique with JPDA-based [44] data association is considered.

  The combination can be based on Gaussian mixture model above the feature space where each of the Gaussians is then considered as a target which is a subject of data association algorithm. It can help to reduce the number of lost object tracks and cluster exchange.

- The object detection technique can be enhanced by object appearance learning in order to support reappearance of the object after the full or partial occlusion.

  For video data, it will mean contribution of object appearance descriptors, based of commonly used feature descriptors like SIFT, SURF, MSER, or any other, in order to model the appearance of the object, which should add to the algorithm the ability to rediscover the objects after their re-appearance.

The TEDA framework can be enhanced by:

- Introduction of statistical tail measures [183] based on TEDA.

  Eccentricity concept can be used to measure the properties of the statistical tails. It can be useful for development of the new statistical criteria.

- Further development of incremental equations based on various distances.

  In this thesis, Euclidean and Mahalanobis [162] distances are discussed. The recursive estimation can also be proven for certain types of Minkowski distance family [204] and the cosine similarity. The extension to some domain-specific, like Levenshtein [205] or Hamming [206], distances.

- Anomaly detection for video analytics based on the TEDA framework.

  Analogously to the problem of classification and clustering, it is possible to state the one-class classification problem and apply it to object of interest detection given the background model.

The image segmentation technique can be extended to multilabel segmentation for better precision and arbitrary segmentation granularity. The changes will involve both the optimisation problem and the numeric scheme.

7 Bibliography

[1] Vision System Design (December 2013) Machine vision: A look into the future http://www.vision-systems.com/articles/print/volume-18/issue-11/features/machine-vision-a-look-into-the-future.html

[2] Ferentinos, K. P. (2005). Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms. *Neural networks*, *18*(7), 934-950

[3] UK Police: Automatic Number Plate Recognition. https://www.police.uk/information-and-advice/automatic-number-plate-recognition/

[4] Railcar number recognition. http://www.axxonsoft.com/integrated_security_solutions/rcnr/

[5] ABBYY® SDK for Developers: Barcode Recognition http://www.abbyy-developers.eu/en:tech:tasks:barcoderecognition

[6] http://www.qrcode.com/en/index.html

[7] http://www.abbyy.com/ocr-sdk-windows/

[8] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (Vol. 1, pp. I-511). IEEE.

[9] Astua, C., Barber, R., Crespo, J., & Jardon, A. (2014). Object detection techniques applied on mobile robot semantic navigation. *Sensors*, *14*(4), 6734-6757.

[10] Hay, G. J., Blaschke, T., Marceau, D. J., & Bouchard, A. (2003). A comparison of three image-object methods for the multiscale analysis of landscape structure. *ISPRS Journal of Photogrammetry and Remote Sensing*, *57*(5), 327-345.

[11] Šuligoj, F., Šekoranja, B., Švaco, M., & Jerbić, B. (2014). Object Tracking with a Multiagent Robot System and a Stereo Vision Camera. *Procedia Engineering*, *69*, 968-973.

[12] Ulrich, I., & Nourbakhsh, I. (2000, July). Appearance-based obstacle detection with monocular color vision. In *AAAI/IAAI* (pp. 866-871).

[13] http://www.controleng.com/single-article/machine-vision-tops-sensors-in-flexibility-for-ford-body-panel-selection/e021aedfb44d48ee7afcf9ef74186cea.html

[14] Borkar, A., Hayes, M., & Smith, M. T. (2012). A novel lane detection system with efficient ground truth generation. *Intelligent Transportation Systems, IEEE Transactions on*, *13*(1), 365-374.

[15] D.Kolev, D. Kangin, G.Markarian (2015). Data Fusion for Unsupervised Video Object Detection, Tracking and Geo-Positioning, Fusion 2015, Washington DC, USA.

[16] D. Kangin, D. Kolev, G.Markarian (2015). Multiple Video Object Tracking Using Variational Inference, Sensor Data Fusion: Trends, Solutions, Applications, 10th Workshop, Bonn, Germany.

[17] P. Angelov, D. Kangin, X. Zhou, D. Kolev (2014), Symbol Recognition with a new Autonomously Evolving Classifier AutoClass, *In Proc. 2014 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS-2014*, 2-4 June, 2014, Linz, Austria

[18] D. Kangin, P. Angelov (2015) Recursive SVM based on TEDA, The Third International Symposium On Learning and Data Sciences

[19] D. Kangin, P. Angelov (2015) Evolving Clustering, Classification and Regression with TEDA, International Joint Conference on Neural Networks, Killarney, Ireland, 2015.

[20] D. Kangin; P. P. Angelov, J. A. Iglesias (2015). Autonomously Evolving Classifier TEDAClass. Journal of Information Sciences.

[21] D. Kangin, P. Angelov, Jose Antonio Iglesias, and Araceli Sanchis (2015). "Evolving Classifier TEDAClass for Big Data." Procedia Computer Science 53 (2015): 9-18.

[22] D. Kangin, D. Kolev, P. Angelov (2016, submitted). Fast Non-parametric Image Segmentation Using Majorisation-Minimisation of a Modified Chan-Vese Functional. International Journal of Intelligent Systems.

[23] D. Kolev, M. Suvorov, and D. Kangin (2016). Kernel models and Support Vector Machines (chapter), P. Angelov (Ed.), *Handbook on Computational Intelligence*, 750pp., World Scientific, accepted,, ISBN: 978-0-470-28719-4

[24] Donoser, M., & Bischof, H. (2006, June). Efficient maximally stable extremal region (MSER) tracking. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (Vol. 1, pp. 553-560). IEEE.

[25] Lombardi, V. C., Ruscetti, F. W., Gupta, J. D., Pfost, M. A., Hagen, K. S., Peterson, D. L., & Mikovits, J. A. (2009). Detection of an infectious retrovirus, XMRV, in blood cells of patients with chronic fatigue syndrome. *Science*, *326*(5952), 585-589.

[26] Kalal, Z., Mikolajczyk, K., & Matas, J. (2012). Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 34*(7), 1409-1422.

[27] Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., & Van Gool, L. (2011). Online multiperson tracking-by-detection from a single, uncalibrated camera. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 33*(9), 1820-1833.

[28] Streit, R., Degen, C., & Koch, W. (2015). The Pointillist Family of Multitarget Tracking Filters. *arXiv preprint arXiv:1505.08000*.

[29] Elgammal, A., Duraiswami, R., Harwood, D., & Davis, L. S. (2002). Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, *90*(7), 1151-1163.

[30] http://www.sesarju.eu/

[31] https://www.faa.gov/nextgen/

[32] Naval Radar Surveillance http://www.terma.com/security-surveillance/radar-systems/naval-radar-surveillance/

[33] Wayne Springer (2011). Positions on the Celestrial Sphere: how to Locate (and Track) Objects from a Spinning, Orbiting Platform In the Space, University of Utah, Lecture Slides http://www.physics.utah.edu/~springer/phys3060/Lectures_files/lec02_2013.pdf

[34] Trucco, E., & Plakas, K. (2006). Video tracking: a concise survey. *Oceanic Engineering, IEEE Journal of*, *31*(2), 520-529.

[35] Särkkä, S. (2013). *Bayesian filtering and smoothing* (No. 3). Cambridge University Press.

[36] Blunsom, P. (2004). Hidden Markov models. *Lecture notes*, *15*, 18-19.

[37] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, *82*(1), 35-45.

[38] G.L. Smith; S.F. Schmidt and L.A. McGee (1962). "Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle" (PDF). National Aeronautics and Space Administration.

[39] Julier, Simon J.; Uhlmann, Jeffrey K. (1997). "A new extension of the Kalman filter to nonlinear systems" (PDF). *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*. Signal Processing, Sensor Fusion, and Target Recognition VI **3**: 182. Bibcode:1997SPIE.3068..182J. doi:10.1117/12.280797. Retrieved 2008-05-03.

[40] Del Moral, Pierre (1996). "Non Linear Filtering: Interacting Particle Solution." (PDF). *Markov Processes and Related Fields* **2** (4): 555–580.

[41] Murphy, K. P. (1998). *Switching Kalman filters* (p. 16). technical report, UC Berkeley.

[42] Zappella, Luca; Lladó, Xavier; Salvi, Joaquim (2008). "Motion Segmentation: a Review". *Proceedings of the 2008 conference on Artificial Intelligence Research and*

Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence Pages 398-407: 398–407.

[43] Reid, D. B. (1979). An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, *24*(6), 843-854.

[44] Fortmann, T. E., Bar-Shalom, Y., & Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association. *Oceanic Engineering, IEEE Journal of*, *8*(3), 173-184.

[45] Mahler, R. P. (2003). Multitarget Bayes filtering via first-order multitarget moments. *Aerospace and Electronic Systems, IEEE Transactions on*, *39*(4), 1152-1178.

[46] Mahler, R.P. (2007, April). A survey of PHD filter and CPHD filter implementations. In *Defense and Security Symposium* (pp. 65670O-65670O). International Society for Optics and Photonics.

[47] Angelov, P., Gude, C., Sadeghi-Tehran, P., & Ivanov, T. (2012, September). ARTOT: Autonomous real-Time object detection and tracking by a moving camera. In *Intelligent Systems (IS), 2012 6th IEEE International Conference* (pp. 446-452). IEEE.

[48] Lowe, David G. (1999). "Object recognition from local scale-invariant features". *Proceedings of the International Conference on Computer Vision* **2**. pp. 1150–1157. doi:10.1109/ICCV.1999.790410

[49] Stratonovich, R.L. (1960). "Conditional Markov Processes". *Theory of Probability and its Applications* **5** (2): 156–178. doi:10.1137/1105015.

[50] Baum, L. E.; Petrie, T. (1966). "Statistical Inference for Probabilistic Functions of Finite State Markov Chains". *The Annals of Mathematical Statistics* **37** (6): 1554–1563. doi:10.1214/aoms/1177699147. Retrieved 28 November 2011.

[51] Subhash Challa, Robin Evans, Mark Morelande and Darko Mušicki (2011): Fundamentals of Object Tracking, Cambridge University Press 2011, ISBN 978 0 521 87628 5.

[52] Pouria Sadeghi-Tehran, Plamen P. Angelov (2014): ATDT: Autonomous Template-Based Detection and Tracking of Objects from Airborne Camera. IEEE Conf. on Intelligent Systems (2) 2014: 555-565

[53] Parzen, E. (1962). "On Estimation of a Probability Density Function and Mode". The Annals of Mathematical Statistics 33 (3): 1065. doi:10.1214/aoms/1177704472. JSTOR 2237880

[54] Kolev, D., Angelov, P., Markarian, G., Suvorov, M. & Lysanov, S. (2013) ARFA: automated real-time flight data analysis using evolving clustering, classifiers

and recursive density estimation, Evolving and Adaptive Intelligent Systems (EAIS), 2013 IEEE Conference on. Piscataway, N.J.: IEEE Press, p. 91-97 7 p.

[55] B.K.P. Horn and B.G. Schunck (1981), "Determining optical flow." *Artificial Intelligence*, vol 17, pp 185–203.

[56] B. D. Lucas and T. Kanade (1981), *An iterative image registration technique with an application to stereo vision.* Proceedings of Imaging Understanding Workshop, pages 121--130

[57] Bouguet, J. Y. (2001). Pyramidal implementation of the affine Lucas-Kanade feature tracker description of the algorithm. *Intel Corporation*, *5*, 1-10.

[58] Farley, B.G.; W.A. Clark (1954). "Simulation of Self-Organizing Systems by Digital Computer". *IRE Transactions on Information Theory* **4** (4): 76–84. doi:10.1109/TIT.1954.1057468.

[59] Zadeh, L. A. (1965). "Fuzzy sets". *Information and Control* **8** (3): 338. doi:10.1016/S0019-9958(65)90241-X

[60] Gibson, J.J. (1950). *The Perception of the Visual World*. Houghton Mifflin.

[61] MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07.

[62] Sobel, I. (2014) History and Definition of the so-called "Sobel Operator", more appropriately named the Sobel-Feldman Operator. https://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator

[63] Prewitt, J. M. (1970). Object enhancement and extraction. Picture processing and Psychopictorics, 10(1), 15-19.

[64] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded up robust features. In *Computer vision–ECCV 2006* (pp. 404-417). Springer Berlin Heidelberg.

[65] Leutenegger, S., Chli, M., & Siegwart, R. Y. (November 2011). BRISK: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (pp. 2548-2555). IEEE.

[66] Matas, J., Chum, O., Urban, M., & Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, *22*(10), 761-767.

[67] Mueller, W. J., & Olson, J. A. (1993, September). Model-based feature extraction. In *Optical Engineering and Photonics in Aerospace Sensing* (pp. 263-272). International Society for Optics and Photonics.

[68] Ranzato, M. A., Huang, F. J., Boureau, Y. L., & LeCun, Y. (2007, June). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (pp. 1-8). IEEE.

[69] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5: 115–133

[70] Hebb, Donald (1949). The Organization of Behavior. New York: Wiley & Sons.

[71] Rosenblatt, Frank (1957), The Perceptron--a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory.

[72] Rosenblatt, Frank (1961) Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC.

[73] B. Widrow and M.E. Hoff, Jr., ``Adaptive Switching Circuits,'' *IRE WESCON Convention Record*, 4:96-104, August 1960.

[74] B. Widrow and M.E. Hoff, Jr., ``Associative Storage and Retrieval of Digital Information in Networks of Adaptive `Neurons,''' *Biological Prototypes and Synthetic Systems*, 1:160, 1962.

[75] von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. Kybernetik, 14:85-100

[76] Kohonen, Teuvo (1982). "Self-Organized Formation of Topologically Correct Feature Maps". *Biological Cybernetics* **43** (1): 59–69.

[77] Fukushima, Kunihiko (1980). "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". *Biological Cybernetics* **36** (4): 193–202. doi:10.1007/BF00344251. PMID 7370364

[78] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

[79] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences of the USA, vol. 79 no. 8 pp. 2554–2558, April 1982.

[80] Tang, Yichuan, Nitish Srivastava, and Ruslan Salakhutdinov. "Learning generative models with visual attention." arXiv preprint arXiv:1312.6110 (2013).

[81] Maas, Wolfgang (1996). "Networks of Spiking Neurons: The Third Generation of Neural Network Models".

[82] Xin Jin; Furber, S. B.; Woods, J. V. (2008). "Efficient modelling of spiking neural networks on a scalable chip multiprocessor". *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. pp. 2812–2819

[83] Kasabov, Nikola K. (2014) "NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data." *Neural Networks* 52 (2014): 62-76

[84] Furber, Steve B. et al. (2014) "The SpiNNaker Project." ?? : 1-14.

[85] Y. Jin (2000). Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement. IEEE Transactions on Fuzzy Systems, 8(2), 212-221, 2000

[86] E. Lughofer (2011). Evolving Fuzzy Systems: Methodologies, Advanced Concepts and Applications. Springer Heidelberg

[87] N. Kasabov (2007). Evolving Connectionist Systems: The Knowledge Engineering Approach - Second Edition. Springer, London

[88] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

[89] Silipo, R., & Marchesi, C. (1998). Artificial neural networks for automatic ECG analysis. *Signal Processing, IEEE Transactions on*, *46*(5), 1417-1425.

[90] Lewis, F. W., Jagannathan, S., & Yesildirak, A. (1998). *Neural network control of robot manipulators and non-linear systems*. CRC Press.

[91] Breiman, L. (1996) Bagging predictors. Machine Learning, **24**, 123-140

[92] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984) Classification and Regression Trees, Wadsworth, Belmont, CA, USA.

[93] Friedman, J. H. (1999). *Stochastic gradient boosting.* Stanford University.

[94] Haar, Alfréd (1910), "Zur Theorie der orthogonalen Funktionensysteme", *Mathematische Annalen* **69** (3): 331–371, doi:10.1007/BF01456326

[95] Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, *14*(771-780), 1612.

[96] Vapnik & Lerner, 1963, Pattern recognition using generalized portrait method, *Automation and Remote Control*, 24, 774–780

[97] Vapnik, V. N., & Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, *16*(2), 264-280.

[98] gBoser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152). ACM.

[99] Basak, D., Pal, S., & Patranabis, D. C. (2007). Support vector regression. *Neural Information Processing-Letters and Reviews*, *11*(10), 203-224.

[100] Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research* (pp. 1453-1484).

[101] Mayoraz, E., & Alpaydin, E. (1999). Support vector machines for multi-class classification. In *Engineering Applications of Bio-Inspired Artificial Neural Networks* (pp. 833-842). Springer Berlin Heidelberg.

[102] Ben-Hur, A., Siegelmann, H. T., Horn, D., & Vapnik, V. (2000, September). A support vector clustering method. In *ICPR* (p. 2724). IEEE.

[103] Schölkopf, B., & Burges, C. J. (1999). *Advances in kernel methods: support vector learning*. MIT press.

[104] Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. *The annals of statistics*, 1171-1220.

[105] Chen, P. H., Lin, C. J., & Schölkopf, B. (2005). A tutorial on ν-support vector machines. *Applied Stochastic Models in Business and Industry*, *21*(2), 111-136.

[106] D. Kolev, M. Suvorov, and D. Kangin (accepted, expected 2016). Kernel models and Support Vector Machines (chapter), P. Angelov (Ed.), *Handbook on Computational Intelligence*, 750pp., World Scientific, ISBN: 978-0-470-28719-4

[107] Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., & Platt, J. C. (1999). Support Vector Method for Novelty Detection. In *NIPS* (Vol. 12, pp. 582-588).

[108] Zadeh, L.A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 3, No. 1, pp. 28-44, Jan. 1973.

[109] Mamdani, E.H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1-13, 1975.

[110] Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *Systems, Man and Cybernetics, IEEE Transactions on*, (1), 116-132.

[111] Angelov, P., & Yager, R. (2012). A new type of simplified fuzzy rule-based system. *International Journal of General Systems*, *41*(2), 163-185.

[112] Angelov, P. P. (2000). Evolving fuzzy rule-based models. *Journal of the Chinese Institute of Industrial Engineers*, *17*(5), 459-468.

[113] D.Kangin, P. Angelov (2015) Evolving Clustering, Classification and Regression with TEDA. International Joint Conference on the Neural Networks, Killarney, Ireland.

[114] Angelov, P. P., & Zhou, X. (2008). Evolving fuzzy-rule-based classifiers from data streams. *Fuzzy Systems, IEEE Transactions on*, *16*(6), 1462-1475.

[115] P. Angelov (2004), “An approach for fuzzy rule-base adaptation using on-line clustering,” Int. J. Approx. Reason., vol. 35, no. 3, pp. 275–289, Mar. 2004

[116] Angelov and D. Filev (2004, February), “An approach to on-line identification of evolving Takagi–Sugeno models,” IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 34, no. 1, pp. 484–498.

[117] Baruah, Rashmi Dutta, and Plamen Angelov (2014). "DEC: Dynamically Evolving Clustering and Its Application to Structure Identification of Evolving Fuzzy Models." *Cybernetics, IEEE Transactions on* 44.9: 1619-1631.

[118] Angelov, P., Dutta Baruah, R., & Andreu, J. (2011). Simpl_eClass: simple potential-free evolving fuzzy rule-based on-line classifiers.

[119] Costa, B. S. J., Angelov, P. P., & Guedes, L. A. (2015). Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier. *Neurocomputing*, *150*, 289-303.

[120] Bishop, C. M. (2006). *Pattern recognition and machine learning. Springer*, pp. 423-455.

[121] Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *22*(8), 888-905.

[122] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing, 17*(4)*,* 395-416.

[123] Chan, T. F., & Vese, L. (2001). Active contours without edges. *Image processing, IEEE transactions on*, *10*(2), 266-277.

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

[125] Nock, R., & Nielsen, F. (2004). Statistical region merging. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *26*(11), 1452-1458.

[126] Ohlander, R., Price, K., & Reddy, D. R. (1978). Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, *8*(3), 313-333.

[127] Tilton, J. C. (1989, July). Image segmentation by iterative parallel region growing and splitting. In *Geoscience and Remote Sensing Symposium, 1989. IGARSS'89. 12th Canadian Symposium on Remote Sensing., 1989 International* (Vol. 4, pp. 2420-2423). IEEE.

[128] Shafarenko, L., Petrou, M., & Kittler, J. (1997). Automatic watershed segmentation of randomly textured color images. *Image Processing, IEEE Transactions on*, *6*(11), 1530-1544.

[129] Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6), 679-698.

[130] Vincent, Luc; Soille, Pierre (June 1991). "Watersheds in digital spaces: an efficient algorithm based on immersion simulations". *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13** (6): 583.

[131] Caselles, V., Catté, F., Coll, T., & Dibos, F. (1993). A geometric model for active contours in image processing. *Numerische mathematik*, *66*(1), 1-31.

[132] Chen, Y., Tagare, H. D., Thiruvenkadam, S., Huang, F., Wilson, D., Gopinath, K. S. & Geiser, E. A. (2002). Using prior shapes in geometric active contours in a variational framework. *International Journal of Computer Vision*, *50*(3), 315-328.

[133] Caselles, V., Kimmel, R., & Sapiro, G. (1997). Geodesic active contours. *International journal of computer vision*, *22*(1), 61-79.

[134] Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, *1*(4), 321-331.

[135] Li, S. Z. (2009). *Markov random field modeling in image analysis*. Springer Science & Business Media.

[136] Pearl, Judea (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Representation and Reasoning Series. San Mateo CA: Morgan Kaufmann. ISBN 0-934613-73-7.

[137] Mumford, D., & Shah, J. (1989). Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, *42*(5), 577-685.

[138] Elboher, E., & Werman, M. (2013). Asymmetric correlation: a noise robust similarity measure for template matching. *Image Processing, IEEE Transactions on*, *22*(8), 3062-3073.

[139] R. Gonzales and R. Woods **(**1992) *Digital Image Processing*, Addison-Wesley Publishing Company, pp 443 - 452.

[140] C.K. Chow and T. Kaneko (1972) Automatic Boundary Detection of the Left Ventricle from Cineangiograms, Comp. Biomed. Res.(5), pp. 388-410.

[141] Hyvärinen, Aapo (1998). "New approximations of differential entropy for independent component analysis and projection pursuit.". *Advances in Neural Information Processing Systems* **10**: 273–279.

[142] Yang, J., Zhang, D., Frangi, A. F., & Yang, J. Y. (2004). Two-dimensional PCA: a new approach to appearance-based face representation and recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *26*(1), 131-137.

[143] Mika, S., Schölkopf, B., Smola, A. J., Müller, K. R., Scholz, M., & Rätsch, G. (1998). Kernel PCA and De-Noising in Feature Spaces. In *NIPS* (Vol. 4, No. 5, p. 7).

[144] Bishop, C. M. (2006). *Pattern recognition and machine learning. Springer*, pp. 561-565.

[145] Bishop, C. M. (1999). Bayesian PCA. *Advances in neural information processing systems*, 382-388.

[146] H. Moravec (1980). "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover". *Tech Report CMU-RI-TR-3 Carnegie-Mellon University, Robotics Institute*.

[147] Lindeberg, Tony (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision* 30(2): 77-116. doi:10.1023/A:1008045108935.

[148] US 2009238460, Ryuji Funayama, Hiromichi Yanagihara, Luc Van Gool, Tinne Tuytelaars, Herbert Bay, "ROBUST INTEREST POINT DETECTOR AND DESCRIPTOR", published 2009-09-24

[149] Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *27*(10), 1615-1630.

[150] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886-893). IEEE.

[151] Gordon, N. (1997). A hybrid bootstrap filter for target tracking in clutter. *Aerospace and Electronic Systems, IEEE Transactions on*, *33*(1), 353-358.

[152] Heeger, D. J., & Jepson, A. D. (1992). Subspace methods for recovering rigid motion I: Algorithm and implementation. *International Journal of Computer Vision*, *7*(2), 95-117.

[153] Stratonovich, R. L. (1960). Conditional Markov processes. *Theory of Probability & Its Applications*, *5*(2), 156-178.

[154] Bishop, C. M. (2006). *Pattern recognition and machine learning. Springer*, pp. 474-485.

[155] Kabsch, Wolfgang, (1976) "A solution for the best rotation to relate two sets of vectors", *Acta Crystallographica* **32**:922. doi:10.1107/S0567739476001873 with a correction in Kabsch, Wolfgang, (1978) "A discussion of the solution for the best rotation to relate two sets of vectors", "Acta Crystallographica", "A34", 827–828 doi:10.1107/S0567739478001680

[156] Leonhard Euler (1776) Formulae generales pro translatione quacunque corporum rigidorum = General formulas for the translation of arbitrary rigid bodies, Novi Commentarii academiae scientiarum Petropolitanae 20, pp. 189–207 (E478)

[157] *Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems,* R-004-1992, ANSI/AIAA, February 1992.

[158] Vincenty, T. (April 1975a). "Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations". Survey Review. XXIII (misprinted as XXII) (176): 88–93.

[159] Angelov, P. (2014). Outside the box: an alternative data analytics framework. *Journal of Automation Mobile Robotics and Intelligent Systems*, *8*(2), 29-35.

[160] Bienaymé I.-J. (1853) Considérations àl'appui de la découverte de Laplace. Comptes Rendus de l'Académie des Sciences 37: 309–324

[161] Tchebichef, P. (1867). "Des valeurs moyennes". *Journal de mathématiques pures et appliquées*. 2 **12**: 177–184.

[162] Mahalanobis, Prasanta Chandra (1936). "On the generalised distance in statistics" (PDF). *Proceedings of the National Institute of Sciences of India* **2** (1): 49–55

[163] Max A. Woodbury (1950), *Inverting modified matrices*, Memorandum Rept. 42, Statistical Research Group, Princeton University, Princeton, NJ

[164] Boykov, Y., & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *26*(9), 1124-1137.

[165] Clifford, P. (1990), "Markov random fields in statistics", in Grimmett, G.R.; Welsh, D.J.A., Disorder in Physical Systems: A Volume in Honour of John M. Hammersley, Oxford University Press, pp. 19–32, ISBN 0-19-853215-6, MR 1064553, retrieved 2009-05-04

[166] Hunter, D.R.; Lange, K. (2000). "Quantile Regression via an MM Algorithm". *Journal of Computational and Graphical Statistics* **9**: 60–77. doi:10.2307/1390613.

[167] Collins, R., Zhou, X., & Teh, S. K. (2005, January). An open source tracking testbed and evaluation web site. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* (pp. 17-24).

[168] Mao, H., Yang, C., Abousleman, G. P., & Si, J. (2014). Automatic detection and tracking of multiple interacting targets from a moving platform. *Optical Engineering*, *53*(1), 013102-013102.

[169] L. Fei-Fei, R. Fergus and P. Perona. *Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories*. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision. 2004

[170] Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *28*(4), 594-611.

[171] Kolmogorov, V., & Zabih, R. (2002). What energy functions can be minimized via graph cuts? In *Computer Vision—ECCV 2002* (pp. 65-81). Springer Berlin Heidelberg.

[172] Y. Wu. Chan Vese Active Contours without edges (2009) http://www.mathworks.com/matlabcentral/exchange/23445-chan-vese-active-contours-without-edges

[173] S.Bagon(2006) Matlab Wrapper for Graph Cut,http://www.wisdom.weizmann.ac.il/ - December 2006

[174] Y.Boykov, O.Veksler, R.Zabih, Efficient Approximate Energy Minimization via Graph Cuts (2001) IEEE transactions on PAMI, vol. 20, no. 12, p. 1222-1239, November 2001.

[175] ETL1 digits database: http://projects.itri.aist.go.jp/etlcdb/etln/etl1/etl1.htm Electrotechnical Laboratory, Japan [In Japanese and English]

[176] Y. LeCun, C.Cortes, C. J.C. Burges. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/

[177] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In *Decision Support Systems*, Elsevier, 47(4):547-553, 2009. http://www3.dsi.uminho.pt/pcortez/wine/

[178] Cortez, P., Teixeira, J., Cerdeira, A., Almeida, F., Matos, T., & Reis, J. (2009, January). Using data mining for wine quality assessment. In *Discovery Science* (pp. 66-79). Springer Berlin Heidelberg.

[179] P. Angelov, Evolving Takagi-Sugeno Fuzzy Systems from Data Streams (eTS+), In Evolving Intelligent Systems: Methodology and Applications (Angelov P., D. Filev, N. Kasabov Eds.), John Willey and Sons, IEEE Press Series on Computational Intelligence, pp. 21-50, ISBN: 978-0-470-28719-4, April 2010.

[180] Zhang, P. (1993). Model selection via multifold cross validation. The Annals of Statistics, 299-313.

[181] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*(3), 273-297.

[182] Oliva, A., & Torralba, A. (2006). Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, *155*, 23-36.

[183] Brys, G., Hubert, M., & Struyf, A. (2006). Robust measures of tail weight. *Computational statistics & data analysis*, *50*(3), 733-759.

[184] Tsai, D., Flagg, M., Nakazawa, A., & Rehg, J. M. (2012). Motion coherent tracking using multi-label MRF optimization. *International journal of computer vision*, *100*(2), 190-202.

[185] Roberts, L. G. (1963). Machine perception of three-dimensional solids. PhD thesis, MIT department of Electrical Engineering.

[186] Fischler, M.A.; Elschlager, R.A. (1973). "The Representation and Matching of Pictorial Structures". *IEEE Transactions on Computers*: 67. doi:10.1109/T-C.1973.223602

[187] http://www.she-philosopher.com/gallery/infotrees_medieval.html

[188] Michael W. Evans (1980) "The Geometry of the Mind." Architectural Association Quarterly 12.4 (1980): 32–55

[189] Wu, C. J. (1983). On the convergence properties of the EM algorithm. *The Annals of statistics*, 95-103.

[190] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" (PDF). Philosophical Magazine 2 (11): 559–572.

[191] C. Harris and M. Stephens (1988). "A combined corner and edge detector" (PDF). Proceedings of the 4th Alvey Vision Conference. pp. 147–151.

[192] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886-893).

[193] D. Gabor, "Theory of communications", Journal IEEE, London, vol. 93, pp. 429–457, 1946.

[194] Abramowitz, M. and I.A. Stegun, *Handbook of Mathematical Functions*, National Bureau of Standards, Applied Math. Series #55, Dover Publications, 1965, sec. 6.5.

[195] Cuturi, M. (2009). Positive definite kernels in machine learning. *arXiv preprint arXiv:0911.5367*.

[196] P. Angelov, Anomalous System State Identification, GB1208542.9, priority date 15 May 2012.

[197] Souza, C. R. (2010). Kernel functions for machine learning applications. *Creative Commons Attribution-Noncommercial-Share Alike*, *3*. http://crsouza.com/2010/03/kernel-functions-for-machine-learning-applications/#cauchy

[198] Poggio, T., & Cauwenberghs, G. (2001). Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, *13*, 409.

[199] L.-J. Li and L. Fei-Fei, "What, where and who? Classifying events by scene and object recognition," IEEE 11th International Conference on Computer Vision, 2007 (ICCV 2007), 2007, pp. 1–8.

[200] Bishop, C. M. (2006). Pattern recognition and machine learning. Springer, pp. 326-344.

[201] Mercer, J. (1909), "Functions of positive and negative type and their connection with the theory of integral equations", Philosophical Transactions of the Royal Society A209 (441–458): 415–446, doi:10.1098/rsta.1909.001.

[202] Kullback, S.; Leibler, R.A. (1951). "On information and sufficiency". Annals of Mathematical Statistics 22 (1): 79–86. doi:10.1214/aoms/1177729694. MR 39968.

[203] Vetrov, D. P., Kropotov, D. A., & Osokin, A. A. (2010). Automatic Determination of the Number of Components in the EM Algorithm of Restoration of a Mixture of Normal Distributions. *Computational Mathematics and Mathematical Physics*, *50*(4), 733-746.

[204] *Minkowski, H.* (1910) Geomeirie der Zahlen, *New York, Chelsea*, reprint, *1953*.

[205] Levenshtein, Vladimir I. (February 1966). "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady 10 (8): 707–710.(In English) = Владимир И. Левенштейн (1965). Двоичные коды с исправлением выпадений, вставок и замещений символов [Binary codes capable of correcting deletions, insertions, and reversals]. Доклады Академий Наук CCCP (in Russian) **163** (4): 845–8.

[206] Hamming, Richard W. (1950), "Error detecting and error correcting codes" (PDF), Bell System Technical Journal 29 (2): 147–160, doi:10.1002/j.1538-7305.1950.tb00463.x, MR 0035935.

[207] P. Angelov, R. Buswell (2002), Identification of Evolving Rule-based Models, IEEE Transactions on Fuzzy Systems, ISSN 1063-6706, vol. 10, No5, pp. 667-677.

[208] Growing Autonomous Mission Management Applications (GAMMA) http://gammaprogramme.co.uk/aboutgamma/

[209] Rinicom Holdings Limited. Patent GB1415372.0 - Object detection. Lodged 29 August 2014.

[210] http://www.iai.uni-bonn.de/~kleind/tracking/

[211] Dominik A. Klein, Dirk Schulz, Simone Frintrop, and Armin B. Cremers (2010, October) Adaptive Real-Time Video-Tracking for Arbitrary Objects, Int. Conf. on Intelligent Robots and Systems (IROS), October 18-22, 2010, Taipei, Taiwan

[212] University of Central Florida. Center for Research in Computer Vision. UCF Aerial Action Data Set http://crcv.ucf.edu/data/UCF-ARG.php

[213] Angelov, P. (2014, December). Anomaly detection based on eccentricity analysis. In *Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on* (pp. 1-8). IEEE.

[214] Rifkin, Ryan, and Aldebaro Klautau. "In defense of one-vs-all classification." *The Journal of Machine Learning Research* 5 (2004): 101-141.