Finding behavioural anomalies in public areas using video surveillance data

Christoffer Brax
School of Humanities and Informatics
University of Skövde
Skövde, Sweden
firstname.lastname@his.se
&
Saab Microwave Systems
Saab AB
Skövde, Sweden

Lars Niklasson
School of Humanities and Informatics
University of Skövde
Skövde, Sweden
firstname.lastname@his.se

Martin Smedberg
Saab Microwave Systems
Saab AB
Gothenburg, Sweden
firstname.lastname@saabgroup.com

Abstract – In this paper we propose an approach for detecting anomalies in data from visual surveillance sensors. The approach includes creating a structure for representing data, building "normal models" by filling the structure with data for the situation at hand, and finally detecting deviations in the data. The approach allows detections based on the incorporation of a priori knowledge about the situation and on data-driven analysis. The main advantages with the approach compared to earlier work is the low computational requirements, iterative update of normal models and a high explainability of found anomalies. The proposed approach is evaluated off-line using real-world data and the results support that the approach could be used to detect anomalies in real-time applications.

Keywords: Anomaly detection, visual surveillance, behaviour modelling.

1 Introduction

In many surveillance missions huge amounts of data need to be gathered, evaluated and analysed in order to make the right decision. Interesting events or threats are often hidden within these large amounts of data. Current surveillance missions typically rely on manual analysis of the data, mainly because it is not feasible to create a general description of "interesting" or anomalous events since this differs between situations. There is an obvious need to develop more intelligent systems with the aim to relieve the users from information overload [1]. These intelligent systems should automatically detect and analyse anomalous events and alert human observers only when appropriate.

Anomaly detection, also known as deviation or outlier detection, has recently gained a lot of attention. In short, anomaly detection is a method to separate a minority of data from a majority of data representing normal events or situations. The separation or detection of the anomalous data is usually guided by a combination of background knowledge and learning from the data itself. The strength of anomaly detection is revealed in real-time applications where large amounts of data have to be analysed on-line. The use of a high-performance anomaly detection system could enable the early proactive detection of potentially threatening events linked to, for instance, terrorism, crime and accidents.

Conventional systems for video surveillance usually rely on manual interpretation of data presented on multiple screens. Although such manual analysis by an experienced analyst can produce good results under ideal operating conditions (few sensors, small area surveillance, low data rate, low requirements on fast response etcetera), manual analysis can also be associated with problems due to the inherent limitations of human analysts such as: inconsistent analysis, missed detections due to distractions, boredom, lack of experience or fatigue, and low throughput during periods of high-volume activity. As a consequence the usage of the large volume of data recorded by extensive networks of cameras operating in public areas such as public transportation facilities, town centres and parking lots has mainly been restricted to forensic purposes.

We propose a method for automating the process of analysing the behaviour of objects in video surveillance. The overall goal with the method is not to replace the operators but to support them in their surveillance mission. This is accomplished by finding objects that deviate from what is considered normal, in the following termed an anomaly. This must be done under run-time conditions and not afterwards to enable the organization to initiate adequate responses to the observed behaviour. One part of such a system is the anomaly detector that uses models of normal behaviours to represent normalcy. These models are then used to find anomalies, i.e. deviations from the normal models, in new observations. In order to build these normal models we must have a suitable representation of object behaviour, i.e. the change in object state over time. There are many ways of

representing object behaviour, ranging from just an ordered list of observations to high level symbols. In addition to this, we also need data in order to capture what is to be considered "normal" for the situation at hand. Since there is no general normalcy model this has to be generated bottom-up from the available data. If a general structure for the representations can be identified, this structure should handle data for various situations and allow the generation of different models of normalcy, depending on the situation.

In this paper we suggest how some important parts of an automatic anomaly detection system can be designed and implemented. The main building blocks are a general structure for representing data, pre-processing of the data, normalcy modelling using situation dependent data and anomaly detection. Our contribution is a set of steps needed to be performed to successfully detect anomalies as well as a proposed state-based representation and normalcy modelling algorithm. Our proposed representation and normalcy modelling method are very simple compared to earlier approaches ([2; 3; 4]), especially the computational complexity generating the normal model. The representation also support iterative updates of the normal model while for example the approach in [3] requires the normal model to be rebuild from scratch to add new observations to the model. The model also have a high explainability to the user, this means that the user can see exactly which states that caused the anomaly. In Section 2 we describe some related work in the area of anomaly detection in visual surveillance and some relevant algorithms for building normal models. The experimental approach is described in Section 3 and results from the experiments can be found in Section 4. In Section 5 we discuss the results and compare them to a previous approach. Finally, some conclusions and future work is found in Section 6.

2 Background

2.1 Anomaly detection

There is no clear definition of anomaly detection within the information fusion community. This is mainly due to the fact that information fusion research in the area of anomaly detection is not as mature as for example object tracking.
Within Computer vision, visual surveillance in dynamic scenes is an active research topic. This topic include applications such as anomaly detection and alarming [5].
In the IT security community, anomaly detection has been used for intrusion detection for a long time [6; 7; 8; 9; 10]. Portnoy et al. [8], defines anomaly detection as:

> "Anomaly detection approaches build models of normal data and then attempts to detect deviations from the normal model in observed data."

The basic idea is to use data regarded as normal to build models representing the normal state in the system. When new data arrives in the system the normal model will be used to classify the new data as normal (the data matches the model) or anomalous (the data does not match the model). An alternative approach to anomaly detection could be to model the minority of data and look for these characteristics in new data; the problem with this approach is that the amount of available data for these kinds of models is often too small and the data is often irregular and difficult to characterize.
Anomaly detection in the area of information fusion can be seen as situation analysis task with the purpose to find objects that in a specific context behave strange compared to the majority of objects. Anomaly detection can be performed on all levels from 0 to 3 in the JDL model [11]. Consider the classical radar detection problem. This problem can be seen as an anomaly detection problem. Given a signal x sampled at specific distance, the detection problem is to decide if there is a target present (hypothesis H1) or not (Hypothesis H0). This can be formulated as:

$$x = s + n \qquad \text{H1: Target present}$$
$$x = n \qquad \text{H0: Target absent}$$

In the formula above, n represents the background noise which can be considered to be normal data, s is the target signal component which representing the anomalous data (occurs rarely). If signal amplitude for a specific range bin exceeds a threshold, a target (anomaly) is detected. If the background noise mostly contains a thermal noise component it can be modelled using a Gaussian with known variance (the thermal noise level of a radar is usually known). This model allows the user of the system to set the threshold based on the false alarm rate. In more complex situations the noise level can be automatically calculated based on collected data by assuming the null hypothesis. This can be seen as training the system on normal data. The threshold is then adaptively set using this information. The main difference between these two approaches is that in the latter case no prior knowledge is required. When a target is detected, the "anomaly" is analyzed to estimate position and speed based on the target signal s.
The simple example above describes the main concepts in anomaly detection, e.g. training the system, representation of normal data, threshold settings and analysis of anomalies. The same concepts apply to anomaly detection in the higher JDL levels. Generally, when climbing in the JDL model, the degree of automation decrease and the need for support from the operator increase.
In this work we are mainly interested in anomaly detection at the situation assessment level (JDL level 2) where the input contains detected objects and their relations to each other and to the environment. The anomaly detection should be able to classify a situation as

normal or anomalous by classifying the objects in the situation as normal or anomalous. The resulting information could be used as input for further processing at JDL level 3 (impact assessment) or for high-lighting the situation to an operator.

2.2 State based anomaly detection

Many of the previous approaches for anomaly detection uses models based on continuous values such as Gaussian mixture models and neural networks.

Many of these only consider momentary observations and it is not obvious how to incorporate the temporal aspects of the data. Yet in many anomaly detection applications it is desirable to find objects that not only deviate in the basic features such as position and velocity, but also objects that deviate in their higher level behaviours such as relations to other objects over time, relations to the environment etc. The first class of anomalies can be regarded as single state anomalies while the other class can be regarded as behaviour anomalies. In this context we consider behaviour as the change in state over time.

When building models of normalcy it is common to extract the basic features in the data such as position and velocity to describe the objects momentary state. An alternative approach is to use a more abstract representation. This can be accomplished by pre-processing the basic features. For example, it is sometimes not that important if an object moves with 12.2 m/s or 12.4 m/s. Instead, it might be more important to know if the object is moving above or below 10 m/s. In this case it might be suitable to create a new feature called SpeedState that can have the discrete values fast or slow that corresponds to a speed above or below 10 m/s. An advantage with this kind of representation is that the object's speed over time can be modelled as a sequence of SpeedStates instead of a sequence of absolute speed values. In this way we can compress the data needed to describe the speed. We can also describe the behaviour of an object by saying: "object X was in (SpeedState=low) for T1 time units and then it changed the speed into (SpeedState=fast) for T2 time units". Various time-series analysing methods could be used to further process the state sequence.

As with the case of the basic feature velocity, the SpeedState can be regarded as a basic or atomic state. If we combine SpeedState with another atomic state, e.g. a discretised position state, we can generate combinations of states that can be handled in the same way as atomic states. The combination of states describes the behaviour of an object on a higher level than the atomic states. We can analyse sequences of atomic- and combination states in the same fashion and for example classify and group objects together. We can also use a database to create probability distributions of states considered normal which can be used to find anomalous behaviours among new observations.

2.3 Related work

Several approaches for anomaly detection concerning deviating motion pattern of vessels in the maritime situation awareness domain have been suggested [2; 3; 4]. In most approaches, an important aspect is the modelling of object behaviour. In [12], a method for clustering vessel trajectories is evaluated. The approach can be used to describe object behaviour at a higher level. This description could be used for normalcy modelling in an anomaly detection application. Bomberger et al. [13] use a neural network approach for describing behaviour patterns of maritime objects. They also use the behavioural models to predict future positions of objects to alert on possible future anomalies. In [3] Gaussian mixture models are used to represent normal behaviour of maritime objects.

Oliver, et al describes a computer vision based system that can model and recognize human behaviours [14]. The system is based on a Bayesian framework with graphical models which can capture both prior knowledge and evidence from data.

Xue and Henderson [15] propose a framework for behaviour analysis. In their experiments they used data from multiple video cameras observing animals, but the framework is general enough to use data from other sensor types. The framework is built upon four modules: behaviour modelling, feature extraction, basic behaviour unit (BBU) discovery, and complex behaviour analysis. There are also several previous approaches for finding anomalies in human activity. Gutchess et al. have built a system called BALDUR (Behaviour Adaptive Learning during Urban Reconnaissance). It learns probabilistic models of observed activity for a given location [16]. The system is based on unsupervised learning techniques and can both learn patterns of normal activity as well as find suspicious or unusual behaviour in real-time. The system uses CCTV cameras and stores all information in a database. The information in the database is mainly used for training the system but can also be used for forensic analysis where the operator can ask the system questions like "Show all pedestrians that approached the entrance door on building X between time T1 and T2". The system was evaluated by using a large dataset containing a month of observed activity from a commercial parking lot. In the evaluation all object positions were discretised into a uniform grid with 3m by 3m cells. The anomaly detection was based on hidden Markov models (HMM) that represented probabilities for normal transitions between cells in a rectangular grid. The result of the evaluation was that the system was able to find anomalies such as cars arriving/departing at unusual hours and pedestrians running in the parking lot.

Duong et al. addressed the problem of learning and recognizing human activities [17]. The domain is activities of daily life with the aim of finding methods to automatically build models of normal activities and their duration and to find anomalous activities based on these

models. The activities are modelled using a two-layered extension of the hidden semi-Markov model. The bottom layer is used to model atomic activities and their duration while the top layer models high level activities that are built from a sequence of atomic activities. The experimental setup is a kitchen with a CCTV camera mounted in each corner, a tracking system producing tracks of the person in the room. The positions are discretised into a 1m by 1m cells. The high-level activities used in the experiments are eating-breakfast, washing-dishes, making-coffee etc. These activities were usually performed in a specific sequence. The system was trained with a number of sequences with typical durations of each high-level activity. In the testing phase they were able to find both normal and anomalous activity sequences as well as finding sequences where the duration of specific activities were deviating from the normal model.

Another approach for finding anomalies in human activity is trajectory clustering. Piciarelli and Foresti [18] use trajectory clustering to learn normal trajectories of cars in a traffic intersection. The input data comes from a multi-camera tracking system. The clustering algorithm used in their experiments cluster trajectories into sub trajectories and then builds a tree of clusters. For each state transition in the tree a probability value is calculated. If a new trajectory has a very low probability of matching the tree of clusters it can be considered anomalous.

Bui et al. [19], used hierarchical HMMs to find anomalous behaviour among people at an airport by analysing transitions between different areas of the airport and mapping them to a number of pre-defined behaviours.

3 Methodology

3.1 Dataset Description

The dataset used in this paper is collected from a video surveillance system installed on top of a building. The surveillance system consists of two camera pairs[1] looking downwards. The left camera pair is directed towards a traffic junction and the right cameras pair mainly covers a small parking lot. The cameras cover an azimuth sector of roughly 90 degrees and the distance to the centre of the scene is approximately 30 meters. The cameras operate in the visual range and we record during day-light only. The cameras are connected to a laptop PC placed inside the building. A view taken from the left and right cameras at the roof is shown in Figure 1. An off-line video tracker was used to extract tracks from the recordings. With some optimizations of the video tracker it should be possible to run it in real-time but in this paper real-time tracking was not the main focus. The video tracker output the following attributes: timestamp, track id, position (x, y), velocity (x, y) and object size. The data-set comes from another research project within SAAB and was reused for the experiments in this paper.

Figure 1: Images from the left and right camera pair.

Normal behaviour of objects in the data

The cameras registered mainly cars, bicycles and pedestrians within the scene. The movements of cars are constrained to the road network. The same applies for the bicycles. The motion of people involves more degrees of freedom but well established walking paths are clearly seen.

The number of established tracks recorded during one day is of the order 1000. We expect that a few days of training is enough to establish a normal model for the scene of interest.

The training data are assumed to be free of anomalies. Obviously, this assumption may be violated when dealing with real-world data. Analysis of the number of false positives in the validation datasets shows that there are a number of anomalies in the training data. However, the number of anomalies is quite small compared to the number of normal observations and therefore they should not have a noticeable impact on the performance of the system.

Anomalies

In order to guarantee some examples of anomalies, we prepared the scene by incorporating a number of anomalous behaviours. Examples of single object anomalies recorded in the test data are illustrated in Figure 2 and include:

1. Person walking on the grass where people normally keep out (1 instance).
2. Person walking on the grass and not on the pavement (2 instances).
3. person running where people usually walk, one instance from left to right and one from right to left (2 instances).
4. Person staying in the scene for a long time (1 instance).

[1] The use of cameras in pairs enable for higher tracking accuracy by using the distance information extracted from the stereo images. The technique is similar to the one used by TRACAB to track football players [18].

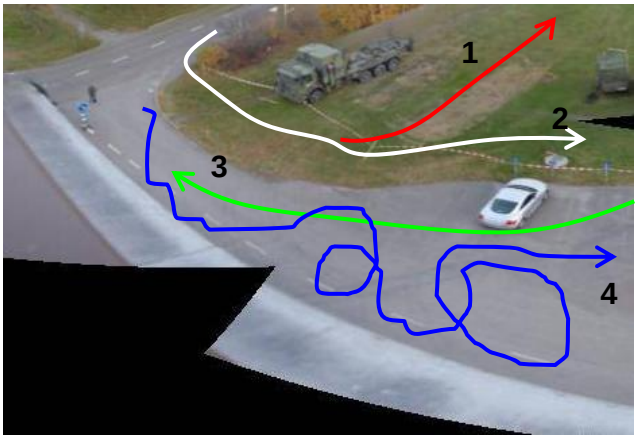

Figure 2: Illustration of single object anomalies.

We also prepared the scene with some examples of complex anomalous behaviour, involving more than one object. These anomalies might require more advanced anomaly detectors to be used, which have it in common that they need to handle more attributes and/or temporal aspects. Examples of complex anomalies include:

- **pick pocketing** scenario involving four persons, where two persons bump into each other, a third picks the wallet from the victim and delivers it to a fourth person (2 instances).
- **snatching**, where a person catches up with a victim, grabs his bag and runs away (2 instances).
- **pursuing**, where a person waits for another person and starts to pursue him when he has passed by (2 instances).

Notice that the scene used in this paper is not very crowded. The pick-pocketing scenario is more likely to occur in a much more crowded area where it would be much harder to detect. A more crowded area would put a lot more stress on the tracking and anomaly detection systems.

3.2 Pre-processing

In anomaly detection and data mining applications, pre-processing is usually the most important step [20]. It is in this step that we find a suitable representation for the data that later will be used by various algorithms. Depending on which algorithm class we want to use we must pre-process the data in different ways. For example, if we use an algorithm that work on discrete data and we have continuous data from our sensors we must discretise the data before it can be used.

In this work we remove the objects, that for some reason (usually by the trackers dead-reckoning algorithms), ends up outside the scene. We also remove all cars in the data by using the *object_size* attribute from the tracker. The main reason for this is that the tracker used in this project was not optimized for tracking cars and hence generated a lot of ghost tracks around the actual cars. Besides the removal of cars, the pre-processing step also includes creating a number of state classes for each observation. The state classes are described in section 3.3.

3.3 State classes

Instead of using the attributes from the tracker we want to use a more abstract or high-level description of the data to be able to incorporate some a priori information about the domain. In this paper we create five basic state classes; *SpeedState*, *CourseState*, *RelationState*, *EnvironmentState* and *PositionState*.

For the *SpeedState* we discretise the speed for each observation of an object into four different classes; Stopped, Slow, Medium and High. The boundary between the classes are chosen to reflect stationary objects, people walking, people running and vehicles respectively. This division assumes that we are interesting in objects which movement could be classified into one of the four classes above. If we would use this representation in another domain, for example maritime security, the boundaries have to be set in another way.

The *CourseState* represents the absolute course of an observation and is divided into eight classes; *north*, *northeast*, *east*, *southeast*, *south*, *southwest*, *west* and *northwest*. Each class is 45 degrees wide. The *north* class is between 337.5 and 22.5 degrees and so on. The course is calculated using the x and y velocities from the tracker.

A *RelationState* defines the relation between one object and other objects in the vicinity. The classes used are *inFront*, *behind*, *left*, *right* and *noRelation*. If the closest distance from an object to another object is above a threshold the *RelationState* is set to *noRelation*. Otherwise the relative position from the object to the closest object is calculated. For example, if an object moves west and another object is east of the first object the relation from the first object to the second one becomes *behind*, etc. The *RelationState* is calculated using the position and course of the objects in the relation.

The *EnvironmentState* describes the relation between an object and the underlying environment. This assumes a priori information about the surveyed area. In our experiments we use the map depicted in Figure 3 to classify the *EnvironmentState* as: onRoad (1), onPavement (2), onGrass (3) or undefined (4).

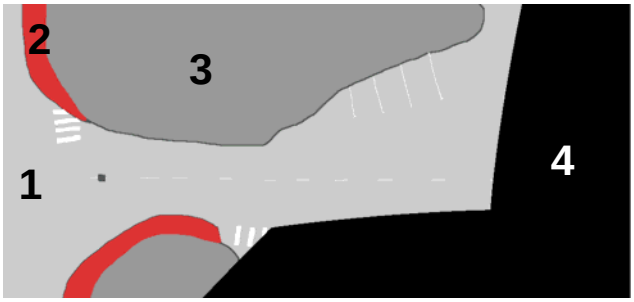

Figure 3: Map used for calculating environment states.

The *EnvironmentState* basically describes on which ground an object is moving and it is calculated by using

the x and y position of the object in conjunction with the map.
The last state class used is the *PositionState* which is a discretisation of the x and y position of an object. In this project we use a 4 row by 10 column grid with a total of 40 different position classes. The size of the grid is decided with respect to the size of the scene. In this case we want the road to be approximately 1 cell wide at the narrow spots. The height and width of the grid cells are constant.
Besides these five atomic state classes we also define a composite state class which is a combination of the five states above. We call this state *CombinationState*, which can be describes as Equation 1.

Equation 1: CombinationState.

$$S_{Combination} = [S_{Speed}, S_{Course}, S_{Relational}, S_{Environment}, S_{Position}]$$

3.4 Training, validation and test dataset

In total we have four days of normal data and one-hour of test data with our 12 anomalies. We use the four days of data for a 4-fold cross validation where data from three days is used as training and the fourth day is used as validation. We call these four sets Setup 1, Setup 2, Setup 3 and Setup 4 (see Table 1 for detailed statistics of the datasets).

	Training dataset		Validation dataset	
	Number of tracks	Number of observations	Number of tracks	Number of observations
1	3720	412592	1824	191977
2	4259	470909	1285	133660
3	4110	438514	1434	166055
4	4543	491692	1001	112877

Table 1: The number of tracks and observations in the four experimental setups.

3.5 Building the normal model

All observations in the training data set are pre-processed which includes the generation of basic and combination states. The resulting dataset is then used to produce the normal model. The normal model is built from a number of combination states (see Equation 1) and their relative frequency in the training dataset. In large datasets with many discrete attributes it might not be possible to find all combination states due to the computationally complexity. In that case, the apriori-algorithm [21] can be used to effectively find the item sets with most occurrences for a dataset. In our case the dataset is quite small and all possible combination rules can be calculated in a matter of seconds. The number of observations in the training data that match a specific combination state gives us a metric for how normal a combination state is. We need at least one observation in the training set to generate a combination state in the normal model. The computational complexity for building the model is O(n). It is also possible to update the model iteratively, without rebuilding it from scratch.

3.6 Anomaly detection

To find anomalies in new observations we search for a combination state in the normal model that matches the new observation. If we find a corresponding state we can look at its relative frequency to get a number on how normal the observation is. If we can not find a corresponding state or if the relative frequency is below a threshold we classify the observation as anomalous. In practise this is done by a simple table-lookup and should therefore be possible to do in real-time for a large number of objects.

4 Experimental results

In this chapter we present the results from our experiments, how the performance is measured and some threshold settings.

4.1 Performance measures and parameter estimation

In 3.6 we described how to classify a single observation in a track either as normal or anomalous. Having a single observation classified as anomalous is clearly not enough for classifying the whole track as normal or anomalous. One or a few observations can be classified as anomalous just due to noise in system. Therefore we need a way to analyse the number of anomalous observations in a track over time. In this paper we use the number of accumulated anomalous observations during the track life-time as an indicator of how anomalous an object is. We also set a threshold for how many anomalous observations that is needed before we classify the whole track as anomalous. The exact value of the threshold is dependent on the application but by analysing the number of anomalous observations for each track in the validation set we can get an indication of suitable values with respect to the number of false positives. In a real system, this value can of course also be set by an operator and could be lowered or raised depending of the current situation.

Assuming a normal distribution of the number of anomalous observations per track, we calculate the mean and standard deviation for all four validation datasets. The result can be seen in Table 2.

Validation dataset	Tracks with anomalies (FP)	Anomalous Observations	Average (mean) number of anomalous observations per track	Standard deviation
Setup 1	14.36%	1.87%	13.67	26.01
Setup 2	8.64%	0.54%	6.56	8.98
Setup 3	16.53%	4.13%	28.95	70.62
Setup 4	14.19%	1.87%	14.88	29.86
Validation dataset 1-4 combined			17.66	45.24

Table 2: The mean and standard deviations calculated for the four validation data sets and for all four sets combined.

As can be seen in table Table 2 the mean and standard deviation for the validation datasets 1 to 4 is 17.66 and 45.24. To get a reasonable number on the threshold for how many anomalous observations a track must have before we classify it as anomalous we use the values from one and two standard deviations in the validation dataset. In this case the thresholds used for detecting anomalies will be 63 (62.91) and 108 (108.15) respectively.
Note that these two thresholds correspond to the object being anomalous for 2.5 and 4.3 seconds (the video tracking system outputs 25 observations per second).

4.2 False positives and detected anomalies

To evaluate the performance of our normal model we use the validation dataset and the tracks from our 12 anomalies. The results can be seen in Table 3. After evaluating our anomalies against the four normal models we found that 37 was the number of anomalous observations from the track with the least amount of anomalous observations, therefore we decided to include this number as a threshold in the evaluation to see the number of false positives when we set the threshold to detect all anomalies.

Threshold	Anomalies found (average)	True positives (average)	False positives (average)
37	12	100.00%	1.35%
63	10.5	87.50%	0.83%
108	6	50.00%	0.49%

Table 3: Results from anomaly detection with the three different thresholds. The results are averages from the four experimental setups.

As expected, the number of false positives increases when we decrease the threshold.

5 Discussion

The results of the experiment are promising, without that much fine tuning and optimization of parameters we manage to find most of the anomalies without a high false positive rate (in the case with the threshold set to 63). We have about seven hours of normal data and in average 12 false alarms each day. This means that our approach in theory would generate less than two false alarms per hour. There were two anomalies that were especially hard to detect, with threshold set to 63 they were only detected in one out of four setups. These two anomalies were the person running where people usually walk (3) from left to right and one instance of the anomaly with a person walking on the grass and not on the pavement (2) which had 37 and 62 anomalous observations respectively. The reason for the problems detecting anomalies with running persons is that they are very similar to people riding bikes which exist in the normal data.

The use of accumulated anomalous observations during the whole track might not be the optimal way of evaluating if a track is anomalous or not. This measure is dependent on the length of the track. If the track is very short it might not have enough observations to be classified as anomalous at all and if the track is very long it might be wrongly classified just because of the fact that it have been in the scene for a long time and gained a lot of anomalous observations. Another approach might be to use a sliding window and count the number of anomalous observations in a shorter time span.
Our approach was compared to a more complex approach involving Guassian Mixture Models (GMM) using the same data. The GMM approach was similar to the one in [3] with the exception of the use of Greedy Expectation Maximization instead of Self Organizing Maps for building the Gaussians. The performance and number of false positives was roughly the same for both approaches. There are several advantages with our propose method compared to the GMM approach. The computational complexity was lower both when building and extending the normal model. Another advantage is that the output from our approach is more intuitive to the user for explaining what caused the anomaly. The user can see exactly in which states that the object deviated from the normal model. In the GMM approach the user has to interpret a number of probabilities for the Gaussian mixtures. We also think that our approach can be modified to also capture temporal aspects while this might be much harder in the GMM approach. There are also some disadvantages with our method compared to the GMM approach. We use apriori knowledge to set the boundaries between the states in the state classes. If these boundaries are set wrongly or not at all the performance of our approach will drastically decrease. The GMM approach does not need this kind of apriori information at all, but to the cost of higher computationally complexity.
The conclusion in this comparison is that is we have some apriori knowledge we can build a much less complex model and still get the same overall performance.

6 Conclusion

In this paper we have presented an approach for intelligent analysis of video surveillance data. We propose a state-based representation of observed objects that allows for incorporation of a priori information about the nature of the expected behaviour. We also present a very simple way of building normalcy models that can be used for detecting anomalies, the approach have low computational complexity and high explainability to the user. The experimental results show that even though the normal model is very simple it can detect most of the anomalies in our test data without a high number of false positives. The performance is in pair with a more complex model based on GMM.

One of the main problems with intelligent video surveillance is to find a spatiotemporal representation for object behaviour. In this paper we mainly considered the spatial part of the representation. In the future we want to extend our normalcy models from just looking on momentary states to sequences of states over time to capture more high level behaviours. We think that the proposed representation can be extended to represent both spatial and temporal aspects of object behaviour. We also want to apply the proposed representation to other domain to test the usefulness in a broader context.

Acknowledgements

This work was supported by the Information Fusion Research Program (www.infofusion.se) at the University of Skövde, Sweden, in partnership with the Swedish Knowledge Foundation under grant 2003/0104, and participating partner companies.

References

[1] S.M. Taylor, How Much Information is Enough?, 10th International Command and Control Research and Technology Symposium, San Diego, CA, USA, 2005.

[2] M. Seibert, B.J. Rhodes, N.A. Bomberger, P.O. Beane, J.J. Sroka, W. Kogel, W. Kreamer, C. Stauffer, L. Kirschner, E. Chalom, M. Bosse, and R. Tillson, SeeCoast port surveillance, Photonics for Port and Harbor Security II. Edited by DeWeert, Michael J.; Saito, Theodore T.; Guthmuller, Harry L.. Proceedings of the SPIE, Volume 6204, pp. 62040B (2006). 2006.

[3] J.B. Kraiman, S.L. Arouh, and M.L. Webb, Automated anomaly detection processor, Enabling Technologies for Simulation Science VI, SPIE, Orlando, FL, USA, 2002, pp. 128-137.

[4] F. Johansson, and G. Falkman, Detection of vessel anomlies - a Bayesian network approach, Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing, 2007.

[5] W. Hu, T. Tan, L. Wang, and S. Maybank, A survey on visual surveillance of object motion and behaviors. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 34 (2004) 334-352.

[6] C. Warrender, S. Forrest, and B.A. Pearlmutter, Detecting Intrusions using System Calls: Alternative Data Models, {IEEE} Symposium on Security and Privacy, 1999, pp. 133-145.

[7] E. Eskin, W. Lee, and S.J. Stolfo, Modeling System Calls for Intrusion Detection with Dynamic Window Sizes, Proceedings of the DARPA Conference and Exposition on Information Survivability. DISCEX '01., IEEE Computer Society, 2001, pp. 165--175.

[8] L. Portnoy, E. Eskin, and S.J. Stolfo, Intrusion detection with unlabeled data using clustering, ACM Workshop on Data Mining Applied to Security (DMSA-2001), USA, 2001.

[9] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L.-W. Chang, A novel anomaly detection scheme based on principal component classifier., IEEE Foundations and New Directions of Data Mining Workshop, 2003.

[10] A. Patcha, and J.-M. Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends. Comput. Networks 51 (2007) 3448-3470.

[11] A.N. Steinberg, C.L. Bowman, and F.E. White, Revisions to the JDL data fusion model. Proceedings of SPIE AeroSense (Sensor Fusion: Architectures, Algorithms and Applications III) (1999) 430-441.

[12] A. Dahlbom, and L. Niklasson, Trajectory Clustering for Coastal Surveillance, Proceedings of the 10th International Conference on Information Fusion, Quebec, Canada, 2007.

[13] N.A. Bomberger, B.J. Rhodes, M. Seibert, and A.M. Waxman, Associative Learning of Vessel Motion Patterns for Maritime Situation Awareness, The 9th International Conference on Information Fusion, Florence, Italy, 2006, pp. 8.

[14] N.M. Oliver, B. Rosario, and A. Pentland, A Bayesian Computer Vision System for Modeling Human Interactions. IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 831-843.

[15] X. Xue, and T. Henderson., Video-based Animal Behavior Analysis From Multiple Cameras, IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Heidelberg, Germany, 2006.

[16] D. Gutchess, N. Checka, and M.S. Snorrason, Learning patterns of human activity for anomaly detection, Intelligent Computing: Theory and Applications V, SPIE, Orlando, FL, USA, 2007, pp. 65600Y-12.

[17] T.V. Duong, H.H. Bui, D.Q. Phung, and S. Venkatesh, Activity recognition and abnormality detection with the switching hidden semi-Markov model, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005, 2005, pp. 838- 845.

[18] C. Piciarelli, and G.L. Foresti, On-line trajectory clustering for anomalous events detection. Pattern Recogn. Lett. 27 (2006) 1835-1842.

[19] H.H. Bui, D.Q. Phung, and S. Venkatesh, Hierarchical hidden markov models with general state hierarchy. in: D.L. McGuinness, and G. Ferguson, (Eds.), The Nineteenth National Conference on Artificial Intelligence, AAAI Press / The MIT Press, San Jose, California, USA, 2004, pp. 324-329.

[20] P.D.P. Edwin, Representation is everything. Commun. ACM 43 (2000) 80-83.

[21] R. Agrawal, and R. Srikant, Fast Algorithms for Mining Association Rules. in: J.B. Bocca, M. Jarke, and C. Zaniolo, (Eds.), Proc. 20th Int. Conf. Very Large Data Bases, {VLDB}, Morgan Kaufmann, 1994, pp. 487--499.