Detecting unsual activities in surveillance video.

By Simon Hartmann Have

Supervisor: Preben Fihl and Thomas Moeslund Date: 31-5-2012

Vision, Graphics and Interactive Systems
Aalborg University

AALBORG UNIVERSITET

Elektroniske Systemer
Institut 8
Fredrik Bajers Vej 7
9220 Aalborg Ø
http://es.aau.dk/

Title:

Detecting unusual activities in surveillance video.

Project period:
September 2011 to May 2012.

Project group:
11gr922

Group members:
Simon Hartmann Have

Supervisor:
Preben Fihl and Thomas Moeslund

Total number of copies: 3

Pages: 89

Finished the 31. May 2012.

Abstract:

This master thesis describes a system for detecting abnormalities in video surveillance by using motion, size, texture and direction features. The method is based on an existing solution, but includes improvements by using a different optical flow algorithm.
The method is tested on the publicly available UCSD anomaly detection dataset with good results within the different categories compared to other methods. Tests have been concluded to view the results of motion, size and texture features independently where the latter have shown to be ineffective.
Two datasets were created for direction based abnormalities. The results on these are very good with an error rate of 0.3% which with small alterations could be used in surveillance systems.
Further research should be put on applicating the methods for video surveillance systems and improving the size and texture feature.

Simon Hartmann Have

Preface

This report describes the development of a master thesis project at Vision Graphics and Interactive Systems Masters programme at Aalborg University.

The report is divided into 5 parts: ”Analysis”, ”Theory”, ”Design and Implementation”, ”Test and results”, ”Conclusion” and ”Future Work”.

In the analysis the current methods and solutions will be reviewed. A problem statement and requirement specification will be presented. The theory chapter will provide solid information on the different methods used in the project.

The design and implementation chapter explains how the theory is used and is separated in the different blocks that the system consists of. This leads up to a test and results chapter where many different methods and parameters are tested to understand which solutions provide the best results.

Finally the conclusion and future work will review to what extend the project fulfils the problem statement and which parts could be further investigated to improve and learn more about abnormality detection.

The project should be understood by people who have a knowledge of computer vision and the different theories that are within this field. The software, images, papers and videos used for this project are enclosed with the report on a CD.

Contents

Chapter 1

Introduction

1.1 Introduction

Imagine a system that takes video from surveillance cameras, understands when something abnormal occurs and relays this information to the relevant user. This would free up the human resources that are now being used to monitor video from these cameras, able cameras that are not being used actively to join in without extra cost and gain much more than a post incident forensic tool. In this project the current solutions available for abnormality detection will be reviewed, state of the art papers will be studied and a method for abnormality detection will be designed, implemented and tested.

Preliminary problem statement:

Is it possible detect abnormal events in video surveillance in real-time?

Chapter 2

Analysis

I Introduction to analysis:
The analysis should provide a solid platform for choosing a method to "Detecting unusual activities in surveillance video". Therefore the history of video surveillance will be reviewed, current solutions and newest research papers will be read and included in a survey. The term "abnormal" will be discussed and defined from a mathematical point of view. A list of possible abnormalities that can be detected will be presented and discussed and finally a problem statement will be formulated. This should constitute enough information to make an informed discussion and end with a requirement specification and list of delimitations.

II Meeting with Milestone system:
This project began with a meeting with Milestone Systems on the 13th of September 2011. Milestone Systems makes platforms for monitoring video surveillance feeds. They expressed a desire to have a system that could detect abnormal activity in video surveillance. The system should be easy to set up, easy to maintain and be able to work in real-time. If the system would take too long to set up, it would be to expensive - especially with many cameras. Exactly the same applies for the maintenance. The real-time requirement is important since this makes it able to detect abnormalities when they occur and not after.

III History of video surveillance:
Video surveillance has been around from before the introduction of the VCR (Video Cassette Recorder) and actually as far back as the birth of the V-2 rocket in Germany in 1942 where it was used for observing launches. The first commercially available camera was introduced by the American company Vericon in 1949 and was advertised to: *"keep an eye on dangerous industrial processes or bring a close-up of demonstrations and surgical operations to large groups of students"*.

The first actual surveillance cameras were installed in the late sixties, start seventies. The police department in New York installed cameras in the Municipal Building and on Time Square in order to prevent criminal activities - unfortunately the crime rates stayed at status quo. Since this was before the introduction of the VCR, the monitors had to be watched at all times.

Only after the VCR became a mass produced and affordable product, did surveillance cameras become a wide-spread phenomenon. Cameras was installed in major London underground stations in 1975, store owners and banks installed cameras for use as evidence after a robbery or other criminal activity. However there were still some major drawbacks using VCRs and analogue technology. The tapes had to be changed or would ware out and the cameras did not perform well in low light or at night. Cameras and technology in general has kept improving with introduction of e.g. the CCD chip, digital multiplexing and motion only recording in the 1990s.

After the terrorist attack on World Trade Center on September 11th 2001, more cameras were installed and more focus was put on processing the surveillance video. It was now time to use the

video, in real time, to detect persons. On site of the Statue of Liberty facial recognition software was installed for terrorist attack prevention, in schools for tracking missing children and sexual offenders [1] and in airports as verification of identity.

In the last few years as computer power has increased and the internet has become accessible from almost anywhere there have been breakthroughs in video surveillance. With good compression algorithms available and increased storage space many days or weeks of video can be saved and more cameras can be connected to the same computer.

IV Increase in cameras:
As video surveillance technology becomes more available and cheaper with the ever increasing fright of terror rises so does the number of cameras. Since there are no requirements to register cameras, there are no certain number of installations.[16] There, however, is no doubt that the number is increasing.

A study conducted by NYCLU (New York Community Board District) [28] showed that from 1998 to 2005 the increase in cameras was more than fivefold, from 769 to 4468. An increase of about 580 % in seven years.

The estimated number of surveillance cameras in the UK is 1.85 million or one for every 32nd person according to [11].

According to [16] this number is even higher in Denmark with an estimated 350.000 cameras or one for every 17th person.

Figure 2.1: Increasing number of surveillance cameras, from [21]

V Effects of video surveillance and impacts on society:
Many studies have been made on the effects of video surveillance, mostly on the subject of crime deterrence. A review of studies was made by "Home Office Research", a British governmental department. In "Crime prevention effects of closed circuit television: a systematic review"[6] 22 study evaluations were included where:

- 11 found a desirable effect on crime.
- 5 found a null effect on crime.
- 6 found a undesirable effect on crime.

The conclusion to the review is that video surveillance only reduces crime to a small degree. Never the less, more and more cameras are installed every day and optimal use of these should always be the goal.

When more surveillance is installed it has an impact on society. In 1999 AC Nielsen AIM did a survey about video surveillance in Denmark, with 994 persons being interviewed[24].

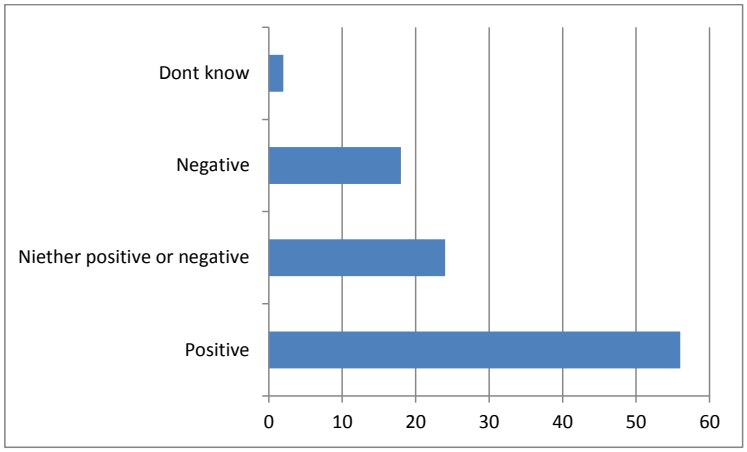


Figure 2.2: Reactions on the increase in video surveillance, from [24]

As it can be seen on figure 2.2 the reactions towards the increase in video surveillance is quite positive with 56% being positive and 24% being neutral. This is question of the general increase in video surveillance.

For some people more surveillance increases their personal security as stated in figure 2.3, but for others it violates their personal space, as stated in figure 2.4.

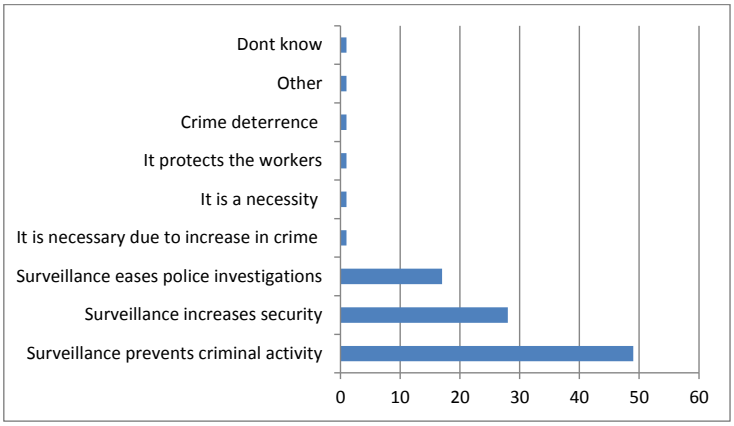


Figure 2.3: Reason for being positive towards video surveillance, from [24]

The primary reasons for being negative towards video surveillance is that it violates the private life of citizens and that people are afraid of a surveillance society. If an automated system could be created where only computers handle the input of these cameras, then people might be less inclined to feel their privacy invaded.

VI Applications of video surveillance:
There are many applications for video surveillance that are not for detecting abnormal activity. In London [2] cameras are used for detecting the license plates of vehicles in specific regions, to check that road pricing fee has been paid. Around the highway tunnel in Aalborg, Denmark, cameras and post processing software detect license plates at different locations and calculate flow of traffic [8]. Video surveillance is also used for remote viewing of dangerous industrial processes

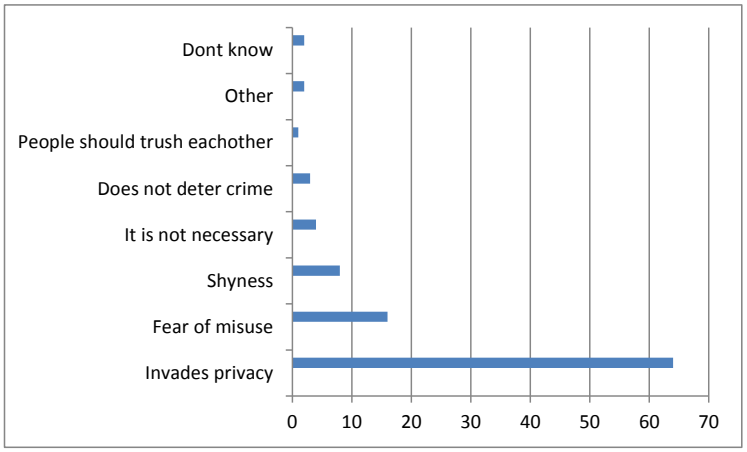


Figure 2.4: Reason for being negative towards video surveillance, from [24]

and numerous other applications.

When it comes to cameras monitoring criminal activity the video can either be used actively to detect happening crimes, accidents or other events of interest or be used post incidental for evidence.

VII Monitoring video surveillance:
For the video to be used actively there must be someone or something monitoring the video to be able to detect abnormal events. This is often done by having multiple screens followed by a human operator who will view these and detect abnormalities. Since humans are very good at perceiving situations this will work as long as the person is able to focus and view all screens within a short while [22] . There is obviously a limit to the amount of screens one person can effectively follow concurrently and as more cameras are installed, more human resources are required. As stated in [26]:

"Mounting video cameras is cheap, but finding available human resources to observe the output is expensive."

VIII Future of intelligent video surveillance:
Automatic abnormality detection will probably not be able to fully compete with actual human beings and our intuitive way of detecting abnormalities any time soon, but it could still be a very helpful tool. As more cameras are installed, more people are required to watch the feed - at least if the purpose is to detect situations when they occur and not as a forensic tool. Hiring more people is expensive and it might not be necessary.

IX Motivation for project:
By developing a system that will give an alarm when something abnormal is occurring it could eliminate the need for extra people to monitor surveillance feeds. Furthermore it might improve the overall performance of the system since the viewer could use the primary focus on watching feeds where there is actually something happening, even if it is not an abnormal event. This would of course depend on how sophisticated the system becomes and which degree of performance it achieves.

As previously mentioned, it might also be able to satisfy the people who have a negative opinion on video surveillance. If it could be used to only save data when something abnormal is occurring

this might reduce their feeling of having their privacy invaded.

2.1 What is abnormal?

What is an abnormality?
For people, abnormal events or occurrences are something that we are learned to identify based on experiences and our ability to understand the context in which things occur. E.g. there is nothing abnormal about a car driving on a road, but a car driving on a playground is highly abnormal. As well as there is nothing abnormal about a child playing on a playground, but it certainly would be if the child was on a road. If we see something we will consider the surroundings, use other senses and several other factors which will help us decide if we should be concerned or not.

An algorithm can be taught what is normal and abnormal to a certain extend, but not to the same extend as humans with years of experiences and a complex brain with many senses - at least not yet. The more complex the input is, the more difficult it becomes to develop very good algorithms for understanding abnormalities.

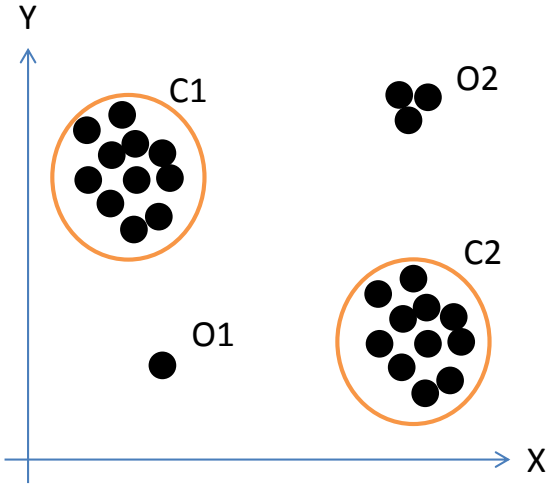


Figure 2.5: Simple example of abnormalities.

For simple input, as visualized in figure 2.5 there are fairly simply solutions to detect abnormalities as O2 and O1. Often identifying outliers is more complex than this and require more sophisticated solutions.

2.2 Mathematical definitions

Mathematical definitions of abnormality (or outliers) is defined by Frank E. Grubbs[15] as:

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

This is a definition by one mathematician, but there is no universal true method for detecting abnormal observations. There are however some model based methods which will be described in the following sections. The models assume normality and univarity in the data.

2.2.1 Three sigma rule[23]

If the input can be considered to be normal distributed the three-sigma rule or empirical rule states that almost all values must lie within three standard deviations σ of the mean μ. As seen in the standardized example in figure 2.6.

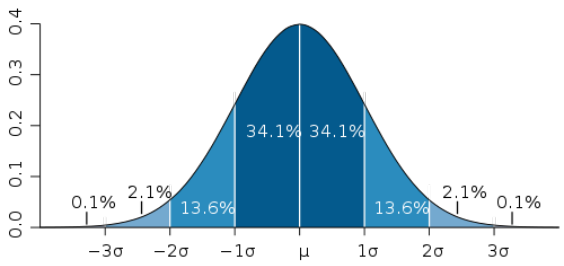


Figure 2.6: Visualization of the three-sigma rule. From [23].

The further away the input is from the normal distribution the more abnormal it is and the three-sigma rule states that 1 in 370 will be more than $\mu \pm 3\sigma$ and 1 in 2149 will be more than $\mu \pm 3.5\sigma$ and so on.

2.2.2 Chauvenet's criterion[4]

Find the standard deviation and the mean of the samples. For each sample find the probability for the sample based on the normal distribution with the standard deviation and mean. Multiply this probability by the number of samples and if this equals a number below 0.5 then reject the sample.

Example: Having measured six samples: $S = [5, 4, 7, 9, 3, 15]$ the mean equals $\mu = 7.16$ and the standard deviation $\sigma = 4.4$.

The last sample 15 differs 1.8σ from the mean which has a probability of 0.07. This probability is then multiplied by the number of observations which gives $0.07 \cdot 6 = 0.42$. Because this value is less than 0.5 the sample should be discarded.

2.3 Time dependency

There are also time depended abnormalities which could be of great use in separating abnormalities from normals depending on the time of day, season, day of the week etc. Congestion on a highway might be very normal in the morning, but maybe not at noon. Another clearly abnormal event would to have snow during the summer season, but this is normal in winter. By adding time dependencies this could improve the abnormality detection.

In figure 2.7 the mean pixel intensity is shown over time. There is a drop in intensity at around 8 o'clock. This would be considered abnormal, but the same intensity would be considered normal 8 hours before.

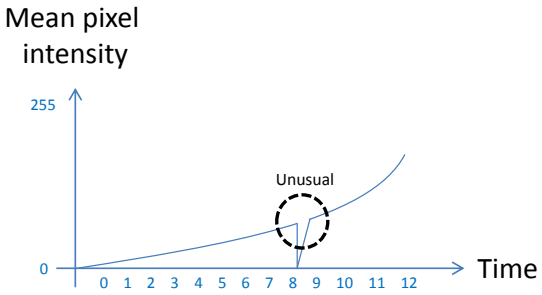


Figure 2.7: Time dependent abnormality.

2.4 Definition of abnormal

In this project abnormal events will be defined as *something that statistically occurs very rarely*. The term ”rarely” is obviously not very well defined and a value for this will have to be determined, either on-line while the program is running or by setting a fixed threshold or probability.

2.5 Detection of abnormalities

To detect *something that statistically occurs very rarely* it is necessary to extract information from the scene and create a model. This model will then be a generalised or average of the normality in the scene.

By using statistical tools or probability theory it is then possible to measure how closely related occurrences in the scene are to the normality. This is known as the classification process which will determine if an occurrence is within the definition of abnormal or not (normal).

Feature extraction:
Extraction of information from a scene is a technique known as feature extraction. A feature, in computer vision, is a mapping of an image or image patch from raw image data to an often smaller representation. A good feature should be transforming the raw image data to a much smaller representation. This should be done by extracting only the key information of the raw data and leaving out the redundant and non describing data. Note that describing data is very different for different feature extraction methods.

Supervised learning:
For generating a model there are primarily two methods; supervised and unsupervised learning.

Supervised learning is a learning algorithm where the user will tell the system which class is being introduced into the classifier. It is also possible to assign weights to represent a cost of classifying an item and set a priori probabilities for these inputs. This method is most applicable to specific situations where the input data is known such as for separating known objects. The major drawback of this method is that it requires a person to manually train the classifier which can be a long and costly process, but for many classification problems this is still very feasible and the achieved results can be very good, especially if the difference between the classes are big.

Unsupervised learning:
Unsupervised learning is the without pre classified data. There is no manual input of what is right and what is wrong. The system must on its own learn from the input and build a classifier that separates one or more classes from each other. The only thing the user has to do is to select the features that should be used for representing the scene and the classifier - the rest should be learned by the program. This could be combined with a method for updating the different parameters and features while the program is running. The major advantage is that the system

does not require any manual input. This significantly reduces setup time of the system and the overall cost of installing it.

2.6 Current methods for detecting abnormalities

When it comes to fixed cameras there are basically two different approaches for detecting abnormalities in video streams: a tracking and a region based. The advantages and disadvantages will be explained in the following sections.

Tracking:
In analysis with tracking, objects are being found in sequential frames and the direction and speed can be computed using e.g. blob tracking, kernel-based tracking or contour tracking [27]. This can then be classified which will indicate if the behaviour is an abnormal. When using tracking methods it is necessary to have a frame-rate that is high enough to ensure that the object will still be in the scene. Furthermore in crowded environments and scenes with much occlusion tracking can be difficult to achieve with good results.

Region-based:
In analysis with non-tracking the scene is split into regions which each are described by features such as SIFT [19], HOG [10] , optical flow [14] , texture analysis with co-ocurrence matrixes or Gabor 2D filters [39] , background to foreground ratio and many more. This technique can be applied to video with low frame-rate since it is not dependent on recognizing the same object temporally.

An example of this can be seen in figure 2.8 which are two consecutive frames from the webcam of the Limfjordsbridge in Aalborg. There are 10 seconds spacing between the frames which makes tracking impossible since the cars could have crossed the bridge faster than this. Therefore it would be better to utilize region-based abnormal detection.

Note that when using optical flow it is crucial that the frame-rate is high enough for the algorithm to compute the flow, which requires the object to remain in the scene. The rest of the features mentioned can be used regardless of frame rate.

Figure 2.8: Region based analysis of scene.

Each region will be described by one or more feature vectors which will be used to train the normal behaviour. When a region is differentiating itself from the normal behaviour of that region, then as well should the feature descriptor differentiate from the normal feature descriptors of that region. This should be recognised by the classifier which outputs an abnormal event.

2.7 Different abnormalities

If it is possible to make a good model for what is normal then it is possible to detect events that does not fit well in this model. These events could be deemed abnormal. To be able to understand how abnormalities could be detected it is important to know which abnormalities that can modelled. A list of abnormalities relevant to this project follows.

Loitering:
When people are standing around the same area for a longer period of time. This is often interesting in places where there normally are few people, or at places where people are for a short period of time. This could be a parking lot, in front of an ATM or other places of interest.

Trajectory based:
In most real life scenes there are paths being used to get from one point to another that appear to be normal. This could e.g. be at a pedestrian crossing or a car driving around on a parking lot.

Direction based:
For one way streets, highways, entrances or exits and all other places where a certain direction pattern occurs. By learning the pattern it should be possible to detect if someone or something is going against the normal flow.

Speed based:
Speed is an important factor in abnormality detection. On a highway the speed is expected to be at least a certain speed, but for abnormal events it could be much lower or much higher. This could be due to queue, accidents or likewise. On a pedestrian street it would be expected to have speeds that equivalents to a person walking. Abnormal events here could be cars, bicycles or people running at speeds that exceed the normal.

Out of place:
If an object that is not normally present in the scene suddenly is, then this could be an abnormality. This could be a car on a pedestrian street or a person walking on a highway.

2.7.1 Possible detection with different methods:

There are some abnormalities that can be detected within the different methods of region- and tracking based. These are listed in the following table.

Method	Loitering	Trajectory	Direction	Speed	Out of place
Tracking	X	X	X	X	
Region			X	X	X

2.8 Current solutions and research

To know the current state-of-the-art research a mini-survey of related articles can be seen in appendix in 10. For easy comparison table 2.1 has been made to illustrate the differences between the methods and to convey which methods could be relevant for this project as described in the forthcoming requirement specification 2.12.

2.9 Current surveillance systems and software

There are many suppliers of surveillance cameras, systems and software so only the a subset of the relevant will be described here. This should give a current overview of what technology is available for purchasing.

Milestone Systems [13]
A Danish company that produces software to operate surveillance cameras. Does not produce

Paper	Speed	Direction	Trajectory	Out of place	Loitering	Background subtraction	Features	Classifier	Learning style	Region/Trajectory	Fps
Mahadevan 2010	x			x		x	MDT	HMM	Supervised	Region	0.04
Reddy 2011	x			x			Optical-Flow, Size, Gabor 2D	PMF, Codebook	Supervised	Region	12
Adam 2008	x	x					Optical-Flow	PDF	Unsupervised	Region	-
Loy 2010	x	x					Optical-Flow	Bayesian	Supervised	Region	-
Breitenstein 2009				x			HoG	Meaningful NN	Unsupervised	Region	5
Zhao 2011	x	x		x			HoG HoF	Dictionary	Unsupervised		0.5
Schuster 2010				x			HoG	Meaningful NN	Unsupervised	Region	20
Ivanov 2009	x	x	x			x	Motion, Color, Texture	SVM	Supervised	Trajectory	-

Table 2.1: Comparison of methods in survey

cameras, but supports all newer models. They offer solutions that integrates surveillance video with other data, such as products being bought at a cash register. Their only vision based application is to recognise license plates for toll booths or at non paying costumers at gas stations.

Axis Communication [3]
A Swedish company operating out of Lund and a producer of network cameras with a global market share of around 30%.

Their own and their partners relevant applications include:

- Video Motion Detection: Records video if motion is detected within a user set polygon.
- Cross Line Detection: Triggers an alarm if a user set line is crossed.
- People Counter: Counts the number of people in a current scene.
- VI-System: Detects left behind objects, loitering, crowding, asset removal and much more.

These applications all require a user to set a polygon or enter a time limit and a camera with high resolution and integrated processor (smart camera) made by Axis.

Geovision [9]
A company founded in Taiwan that produces their own line of 37 different cameras.

Their relevant applications include:

- Object tracking: Working with both normal and fish-lens cameras.
- Face detection: Detect human faces and saves instantly as jpeg for further detection by other software.
- Crowd detection: Detects crowds that are bigger than a specific threshold in a user set polygon.

These applications also require a camera made by GeoVision to work.

Summery
There are several other companies that have applications for recognition of license plates, faces, line crossing, motion detection or crowd detection. These technologies are well established and works with great efficiency, especially with good cameras and light conditions. They do not require much interaction from users apart from a small setup which could be setting a polygon of the area of interest or a threshold.

These applications are great for what they are designed for, but they do not have the capability of detecting abnormal events in general. There is nothing intelligent about them and they do not learn anything from the occurrences in the scene.

2.10 Discussion of articles and current solution

One of the most important aspects about a system for abnormality detection should be that it can be used in a real-time. If this is not possible it will become a forensic tool that would be used after a crime or similar have occurred. Some of the reviewed papers do not state what rate their algorithm runs with, but the two fastest proposed by Schuster et al and Reedy et al. reach an acceptable 20 [47] and 12 fps [45].

Another important aspect is which abnormalities the algorithm can detect. For the system to be somewhat robust towards detecting abnormalities it should be able to detect more than just one of the abnormalities listed in 2.7. The method proposed by Schuster et al[47] utilises the ”histogram of oriented gradients” within regions and can only detect when objects are out of place. The method proposed by Reddy et al [45] fulfils the real-time requirement while also being able to detect two different types of abnormalities, namely the motion and out place abnormalities.

The different methods work either both supervised or unsupervised learning. A supervised approach for training a classifier can be feasible for very specific tasks such as object classification, but for abnormality detection on more than one camera the training phase could be very time consuming. Especially if the system should work on a large surveillance camera ring. Therefore the system should be able to learn on itself and furthermore should be able to adapt to changes in the scene.

2.11 Problem statement

With the knowledge of current commercial solutions and papers the problem statement can be made.

Is it possible to robustly and automatically detect abnormal events in video surveillance in real-time where the abnormal events are speed, direction and out-of-place objects?

2.12 Requirement specification and delimitation

The requirements specification is based on the initial talk with Milestones Systems and an article by Adam et al[29] from where it has been chosen to adopt their specification, since they cover almost all needs. The requirements should clarify what the system should be able to do, but also help in the development phase.

Requirement specification from [29] with the last requirement added for this project.

- Tuning the algorithm for a given video stream should be simple and fast — this process is done by technicians and for many dozens of cameras at every installation site.
- The algorithm should be adaptive - the environment may change (different times of day for example).
- Should be able to run in real-time.
- Robustness - many real-life scenes are crowded and cluttered.
- Fully automatic learning (unsupervised) and operation, with a (desirably) short learning period.
- Low computational requirements - either the algorithm will run on a DSP card installed with the camera ("smart camera") or on a small number of PCs handling multiple cameras. It is unreasonable to require a PC for every camera.
- Predictable performance - which should be adequate in many operationally interesting scenes.
- Should be able to detect more than one abnormality as those listed in section 2.7.

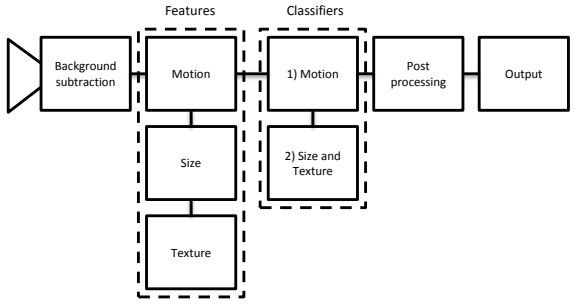
Delimitations:

- The camera will be fixed without the possibility of zoom.

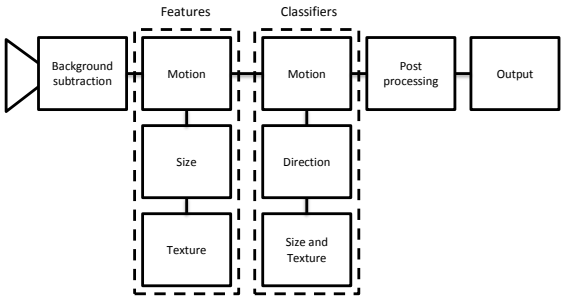
2.13 Chosen method

fAs it can be seen in table 2.7 there are none of the reviewed papers that fulfil the requirement specification. The method proposed by Reddy et al. "Improved anomaly detection in crowded scenes via cell-based analysis of foreground speed, size and texture." [45] can detect speed and out-of-place objects and also fulfils the real-time requirement, but does not fulfil the requirement of being able to detect direction based abnormalities. This is a solid starting point with fulfilling three out of four important requirements and to not start from nothing, the method will be the starting point of this project.

In figure 2.9a the method used in [45] is described in blocks. By expanding the method to also use direction the method implemented in this project will be as in figure 2.9b. Notice that there are not more feature blocks, but one extra classifier. That is because both speed and direction can be computed using the motion features, but in [45] only the gradient of the motion (speed) was used. Therefore including direction should not require much computational power and by that still uphold the requirement of real-time.



(a)



(b)

Figure 2.9: The original and expanded system.

Tests will be conducted more thoroughly and the motion, direction and size/texture classifiers will be tested independently, to gain knowledge of their individual performance. The test for motion, size and texture will be done on the UCSD anomaly dataset since other comparable methods have also used this. For the direction abnormalities a real-life and synthetic dataset will be created by the author and tested upon.

Tests which this project will focus on includes:

- Motion abnormalities: Three different optical flow algorithms.
- Motion abnormalities: The effect of region size.
- Motion abnormalities: Number of histogram bins for the classifier.
- Motion abnormalities: The effect of kernel density estimation.
- Direction abnormalities: Three different optical flow algorithms.
- Direction abnormalities: The effect of region size.
- Direction abnormalities: Incrementing method.
- Computational requirements.
- Pixel-wise abnormality location.
- Unsupervised vs. supervised training of the classifier.
- Effects on post-processing.

Chapter 3

Theory

3.1 Introduction to theory

In this part the theory that will be used in the project is explained and discussed. This should provide enough information to make the design of the system on a qualified background.

The following subjects will be presented and explained:

- Background subtraction.
- Motion: Optical flow.
- Texture: Gabor filter.
- Probability mass function from samples.
- Kernel density estimation.

3.2 Background subtraction

To lower computation and to only take foreground into account it is necessary to perform background subtraction, or in another word, foreground segmentation. There are several methods to do this and many of them have been used intensively in the computer vision. Therefore there are many surveys, comparisons and new methods prosed.

Requirements for background subtraction method:

- Should be able to perform in real-time.
- Should be reliable.
- Should be adaptive to changes in the scene.

All of these methods work on a per pixel level and it is therefore important that the camera is static, as described in the requirement specification 2.12. The focus on this project is not to extensively understand and research background subtraction methods, therefore it has been chosen to only focus on the methods currently implemented in the OpenCV framework [31] which contains the MOG [37] and FGD [40]

3.3 Motion: Optical flow

Optical flow is the apparent motion of the objects from one frame to the next frame, relative to the observer. It is possible to use methods for obtaining an optical flow field, but this is not the same as obtaining the actual motion in the scene. There is a difference between an optical flow field and a motion field as illustrated in figure 3.1.

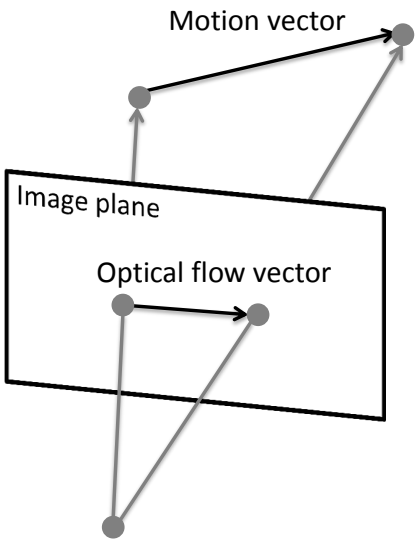


Figure 3.1: The difference between optical flow field and the real motion field.

The optical of a scene will always be an approximation of the real motion, since there is a mapping from a 3D environment to a 2D surface.

Optical flow is often done on pixel level and will result in a flow field which is a field of vectors that could be used to transform one frame into the next one. Optical flow can in this project both be used to describe direction and speed, two descriptors that are very useful for describing the behaviour in a scene.

For estimating optical flow there are three major methods where on is a sub method: Lucas-Kanade[42], Pyramid Lucas-Kanade[30] and Horn–Schunck[35] which will be described in the following sections.

Brightness constancy equation:

Many of the methods for computing optical-flow rely on the brightness constancy equation which assumes that the pixels do not change intensity when they move from frame to frame. The assumption introduces equation 3.1 and the idea is illustrated in figure 3.2. Note that the ”brightness constancy equation” can be refereed to as ”color constancy equation” when dealing with color images.

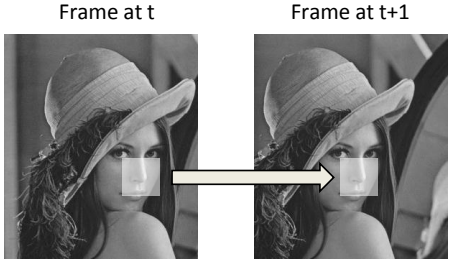


Figure 3.2: The brightness constancy within a neighbourhood is constant from one frame to the next frame.

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{3.1}$$

Where I is the intensity and Δ is the movement.

By first order Taylor expansion that is:

$$I(x, y, t) \approx I(x_{t_o}, y_{t_o}, t_{t_o}) + \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t}\frac{\partial t}{\partial t} \tag{3.2}$$

The brightness constancy assumption can be now again be used to simplify the expression:

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t}\frac{\partial t}{\partial t} = 0 \tag{3.3}$$

Which can be rewritten to:

$$I_x u + I_y v + I_t = 0 \tag{3.4}$$

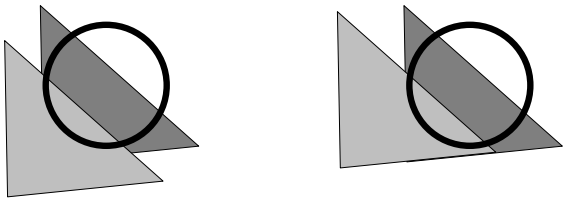
$$I_x u + I_y v = -I_t \tag{3.5}$$

Where I_x and I_y are the image derivatives and I_t is the difference between the frames.

This equation cannot be solved since there is only one equation, but two unknowns. This is known as the "Aperture problem".

Aperture problem:

A aperture problem arises because the movement seen through the aperture is ambiguous as illustrated in figure 3.3. Even though the displacement from the dark to light triangle is different, it is seen through the aperture to be the same.



(a) Displacement 1: from dark to light triangle.

(b) Displacement 2: from dark to light triangle.

Figure 3.3: Visualisation of aperture problem.

There are several solutions to this problem which will be described in the following sections.

3.3.1 Lucas-Kanade method[42]:

The method was developed by Bruce D. Lucas and Takeo Kanade in 1981. The following assumptions are made:

- Assumes the brightness constancy equation.
- That there is spatial coherence between frames.
- That the motion in a small neighbourhood is equivalent.

To solve the aperture problem the method assumes that the motion in a small neighbourhoods is equivalent e.i. u, v are constant in a window of m x m. This will lead to an over-determined

system:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_n) \end{bmatrix} \tag{3.6}$$

Or simplified:

$$A\vec{x} = -b \tag{3.7}$$

An over-determined system can be solved using least squares:

$$A^T A\vec{x} = A^T b \tag{3.8}$$

$$\vec{x} = (A^T A)^{-1} A^T b \tag{3.9}$$

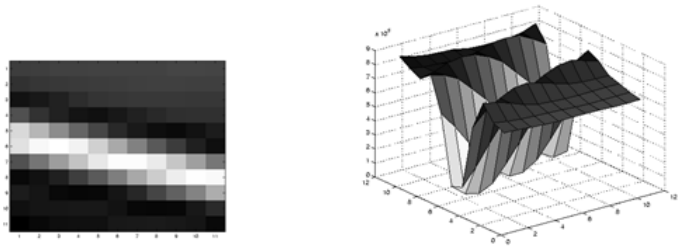
Which sum up to:

$$\begin{bmatrix} u \\ v \end{bmatrix} \begin{bmatrix} \sum_i I_x(p_i)^2 & \sum_i I_x(p_i) I_y(p_i) \\ \sum_i I_x(p_i) I_y(p_i) & \sum_i I_y(p_i)^2 \end{bmatrix}^{-1} = \begin{bmatrix} -\sum_i I_x(p_i) I_t(p_i) \\ -\sum_i I_y(p_i) I_t(p_i) \end{bmatrix} \tag{3.10}$$

Where i runs from 1 to n.

There are a few problems with the standard Lucas-Kanade method. To be able to solve the least squares problem we need for the matrix $A^T A$ to be invertible, i.e. have non-zero eigenvalues. To understand this better some of the situations where the matrix has close to zero, different or high eigenvalues.

Edge:
An motion along an edge will be ambiguous as explained in the aperture problem section 3.3. This would result in $A^T A$ becoming singular and therefore not invertible.



(a) Close up of edge in an image.

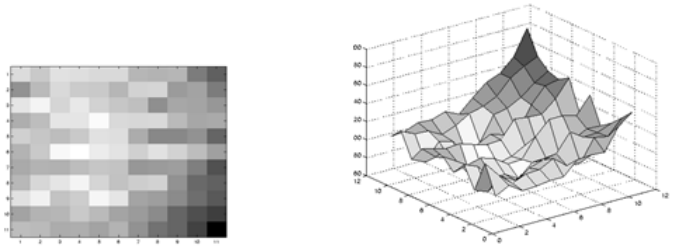
(b) Contour plot of edge.

Figure 3.4: Edge image and contour plot, from [18]

The example in figure 3.4 will produce big and small eigenvalues, respectively for the gradient along the edge and the gradient away from the edge.

Homogeneous area:
In an area where every pixel has same intensity derivative in both directions will be close to zero

which will make every scalar in the $A^T A$ matrix zero. This will result in eigenvalues of values close to zero and the matrix will not be invertible.



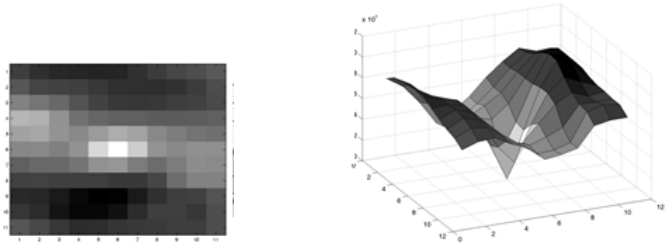
(a) Close up of low texture region in image.

(b) Contour plot of low texture region.

Figure 3.5: Edge image and contour plot, from [18]

The example in figure 3.5 will produce small eigenvalues for both gradients and the area will not be suitable for optical flow estimation.

Textured regions:
In textured regions with much variance the intensity derivative will not be zero and it will not be a problem to invert the matrix. Therefore textured regions are considered as a good place to measure optical flow with the Lucas-Kanade method.



(a) Close up of high texture region in image.

(b) Contour plot of high texture region.

Figure 3.6: Edge image and contour plot, from [18]

The example in 3.6 will produce high eigenvalues since the gradients both have high magnitude. This will be a suitable area for tracking. A method for finding these areas will be introduced in the following section.

Good features to track

A method often named as "Shi-Tomasi corner detector" was introduced in "Good features to track"[48] by Jianbo Shi and Carlo Tomasi in 1994. It uses eigenvalues to obtain places of corners, which are the optimal places to calculate the optical flow. These points are found at corners by using a method based on the Harris Corner detector, but with different criteria for evaluating the found eigenvalues.

The Harris corner detector works by finding the SSD (Sum of Squared Differences) between an

image patch and a patch defined by displacement (u, v):

$$SSD(u,v) = \sum_x \sum_y w(x,y)[I(x+u, y+v) - I(x,y)]^2 \tag{3.11}$$

Where I is the intensity and $w(u, v)$ is a window function.

For constant areas the term in the square brackets will be low and for areas with high variance the value will be high. Therefore the areas where $SSD(u, v)$ is high are of great interests.

The term in equation 3.11 can be rewritten to the following equation using Taylor expansion:

$$SSD(u,v) = [u,v]A\begin{bmatrix} u \\ v \end{bmatrix} \tag{3.12}$$

Where A is:

$$A = \sum_x \sum_y w(x,y)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \tag{3.13}$$

The corner criterion is for the Harris corner detector:

$$det(M) - k(traceM)^2 > T \tag{3.14}$$

And for the Shi-Tomasi:

$$min(\lambda_1, \lambda_2) > T \tag{3.15}$$

Where k is an empirically determined value between 0.04-0.15 [5] and T is a threshold.

Where the Harris corner detector does not have to compute the eigenvalues, but only the determinant and trace of the matrix, the Shi-Tomasi has to compute these values. This requires extra computation, but the results are also greater according to [33].

By using "good features to track" together with standard Lucas-Kanade it should be possible to minimise the errors due to the aperture problem. There is however still issues when the displacement is too big for the search area.

3.3.2 Pyramid Lucas Kanade [30]:

Standard Lucas-Kanade works well for small displacements, but if the displacement is not small it can be a possibility to use an extension of the original method which uses pyramids. This will lower the images resolution where the displacement will be smaller and after calculation of the flow the image can be warped and upsampled. This method was introduced in "Pyramidal Implementation of the Lucas Kanade Feature Tracker" [30] and is implemented in OpenCv[31]. An illustration of the method can be seen in figure 3.7.

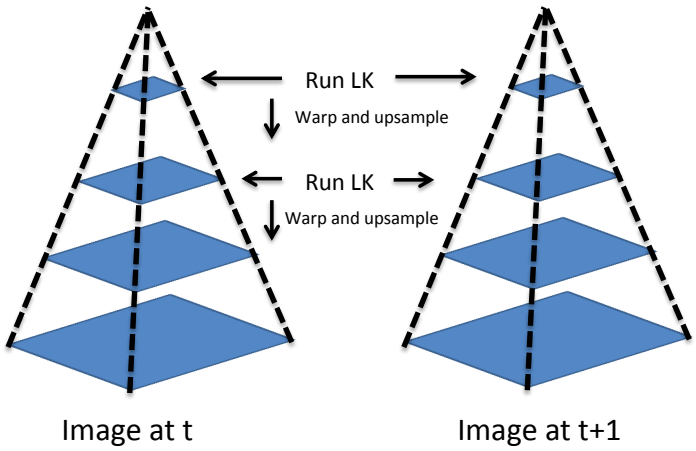


Figure 3.7: Lucas-Kanade method with pyramids.

The method in OpenCV is implemented as Lucas-Kanade optical flow algorithm with pyramidal approach and the use of "good features to track".

3.3.3 Horn-Schunck method[35]:

A method that can efficiently compute dense flow fields is the Horn-Schunck method which was developed by Berthold Horn and Brian Schunck in 1980. The following assumptions are made:

- Assumes the brightness constancy equation.
- That the flow-field is globally smooth.
- Neighbouring pixels have almost the same velocity.

The energy functional used is, where the first part is derived from the brightness constancy equation and the last from the global smoothness assumption:

$$E = \int\int [(I_x u + I_y v + I_t)^2 + \alpha^2(\|\nabla u\|^2 + \|\nabla v\|^2)]dxdy \tag{3.16}$$

I_x, I_y, I_t are the derivatives of image intensity and α is the regularization constant where larger α leads to a smoother motion flow. $|\delta u| = \sqrt{u_x^2 + u_y^2}$ and $|\delta v| = \sqrt{v_x^2 + v_y^2}$ and u, v is the optical flow vector.

The energy functional 3.16 should be minimized by finding appropriate values for the optical flow vector(u,v). This is done by solving the Euler-Lagrange equations which are used to solve differential equations that are stationary (e.i. derivative is zero) at their extrema. This gives the following equations:

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \Delta u = 0 \tag{3.17}$$

$$I_x(I_x u + I_y v + I_t) - \alpha^2 \Delta v = 0 \tag{3.18}$$

Where Δ is the spatial Laplace operator $\Delta = \partial_{xx} + \partial_{yy}$. The two equations can be solved simultaneously by using the following iterative solution:

$$u^{k+1} = \hat{u}^k - \frac{I_x(I_x\hat{u}^k I_y\hat{v}^k + I_t)}{\alpha^2 I_x^2 I_y^2} \tag{3.19}$$

$$v^{k+1} = \hat{v}^k - \frac{I_y(I_x\hat{u}^k I_y\hat{v}^k + I_t)}{\alpha^2 I_x^2 I_y^2} \tag{3.20}$$

Comparison of methods:

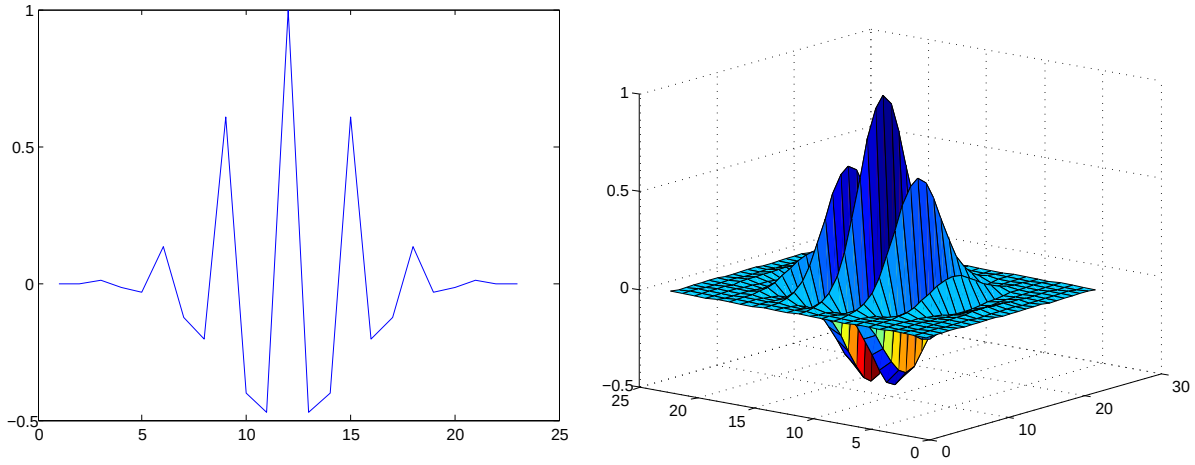
Where Lucas-Kanade is a local method which operates on small regions to obtain the optical flow, Horn-Schunk operates as a global method that uses the global smoothness to compute optical flow. This will of course have differences in the outcome of the optical flow field. How this will impact the results will be evaluated in the test chapter 5.

If the wish is to track only a few selected pixels (by e.g. using "Good features to track") an optimal method could be the Lucas-Kanade combined with Gaussian pyramid implementation, which is one of the methods included in OpenCV.

3.4 Texture: Gabor filters

The descriptor for texture is calculated on the basis of a Gabor filter convoluted with the input image in four different directions.

The Gabor [39] filter is linear filter often used in computer vision in describing texture, finding edges, image coding and various other subjects. The filter has received much attention especially since it was discovered that some cells, responsible for processing visual information in mammals, could be described by these filters. In the spatial domain a Gabor filter is a sinusoidal wave modulation of a Gaussian kernel.



(a) 1D Gabor filter.

(b) 2D Gabor filter.

Figure 3.8: Gabor filter visualised in 1D and 2D.

Definition

The filter is composted of, for this use, a real part which is given this definition in [39]. There are other definitions which slightly different parameters, but this was chosen due to its simplicity in range of variable parameters.

$$G(x;y;\omega;\theta) = \frac{\omega}{\sqrt{2\pi}K} e^{-\frac{\omega^2}{8K^2}(4x^{'2}+y^{'2})}[\cos(\omega x^{'}) - e^{\frac{-k^2}{2}}] \tag{3.21}$$

$$x^{'} = x\cos\theta + y\sin\theta \tag{3.22}$$

$$y^{'} = -x\sin\theta + y\cos\theta \tag{3.23}$$

Variables of Gabor filters

- Wavelength ω: The radial frequency of the filter.
- Orientation θ: Specifies the normal vectors orientation to the parallel lines of the filter.
- K-factor: K : The frequency bandwidth, where one octave is approximately π.

Magnitude of filter output

Obtaining the magnitude is done by taking the absolute value of the output.

$$G_{magnitude} = |G| \tag{3.24}$$

Example of convolution

To illustrate the concept behind the Gabor filters and the utilization in this project the following image 3.9 will be processed by a Gabor filter in the four directions θ of 0°, 45°, 90°and 135°.

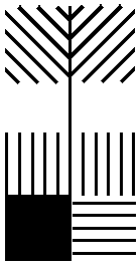
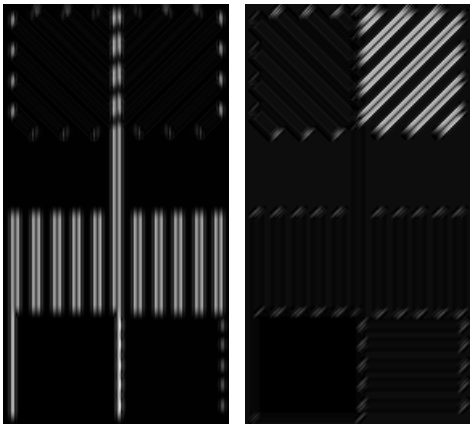


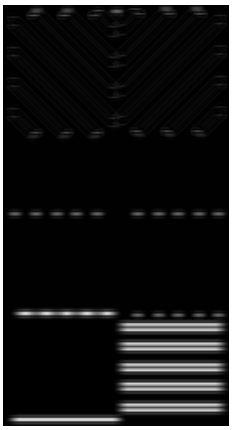
Figure 3.9

The parameters was set to a kernel size of 7 x 7, $\omega = 2.3$, $K = \pi$ and gives the output which can be seen in figure 3.10.



(a) 0 °

(b) 45 °



(c) 90 °

(d) 135 °

Figure 3.10: Gabor filter output in four directions, for the image in 3.9

As it can be seen in figure 3.10 the lines which are oriented in the specified direction are detected.

3.5 Probability mass function (pmf) from samples

To know if data from a certain feature (velocity, direction, intensity etc.) is abnormal or not a probability for this must be calculated. With sample feature data, this can be turned in to a histogram and discretized to a probability mass function (pmf). This is in sense a probability density function (pdf) with discrete values and the possibility of finding a probability without integrating over an interval.

For a pmf ($f(x)$) the following equation must be fulfilled:

$$\sum_{x=1}^{N} f(x) = 1 \tag{3.25}$$

Data samples of certain features can be made in to a histogram by binning the samples from the minimum to maximum value for a fixed number of bins. After the values must be normalized such that equation 3.25 is fulfilled. This can be done by dividing every value with the sum of all

values as in equation 3.26.

$$f_{pmf}(x_0) = \frac{f_h(x_0)}{\sum_{x=1}^{N} f(x)} \tag{3.26}$$

Where $f_h(x)$ is an entry in the histogram.

To overcome the binning problem where the generated distribution does not match the underlying distribution, the histogram will can be smoothed using kernel density estimation.

3.6 Kernel density estimation

Kernel density estimation is a method for overcoming the problem of trying to model a population that is based on a finite sample. When making a coarse histogram there can be a great deal of the data that lies very close between two bins. This could cause the histogram to represent the underlying population badly.

By using kernel density estimation the data is smoothed by a kernel which takes the neighbouring bins into consideration. This can be done by using different types of kernels. First the kernel density estimator is defined as:

$$\hat{f}_h(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \tag{3.27}$$

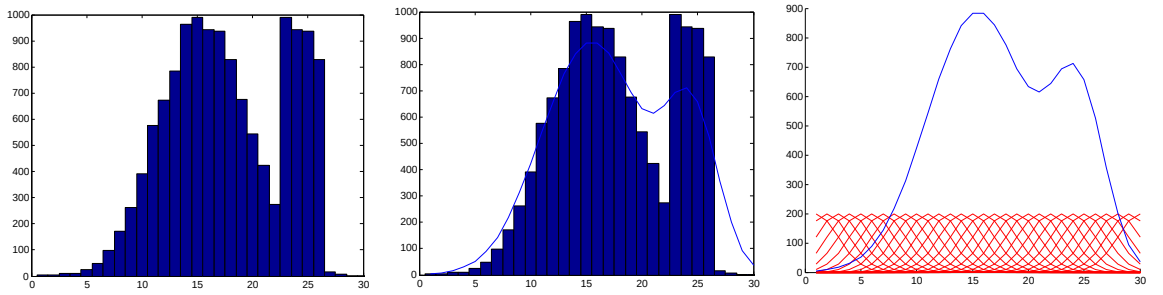
Where N is the number of samples, h is the bandwidth, x is the input variable and x_i is x at the i'th position.

There are many different types of kernels such as Uniform, Triangular, Epanechnikov, Triweight and several others. The most commonly used is the Gaussian kernel where the bell-shaped probability density functions acts as kernel for each point. Using the kernel density estimator together with a Gaussian kernel produces the following equation:

$$\hat{f}_h(x) = \frac{1}{Nh\sqrt{2\pi}} \sum_{i=1}^{N} f(x_i) e^{\left(-\frac{x-i}{2h^2}\right)} \tag{3.28}$$

Where N is the number of samples, h is the bandwidth, x is the input variable and $f(x_i)$ is the value of the i'th variable.

For each bin in the histogram the new estimated value is computed based on the Gaussian kernel. An example of this can be seen in figure 3.11



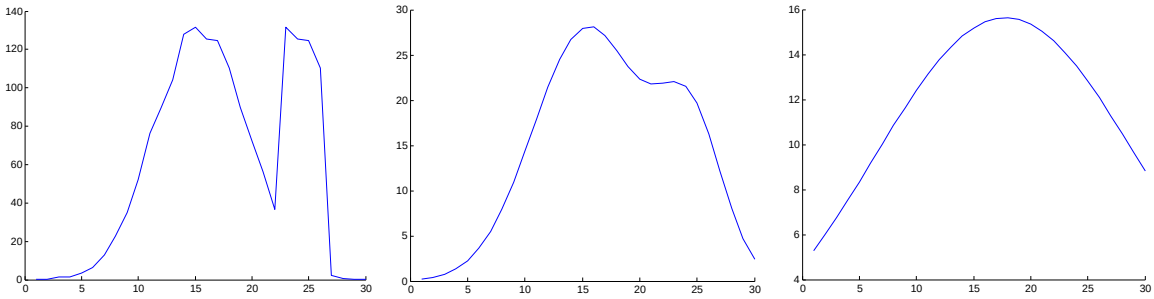
(a) Semi binomial randomly generated histogram.

(b) Estimated distribution.

(c) Estimated distribution with Gaussian kernels (in red) per estimated point.

Figure 3.11: Example of Kernel density estimation with Gaussian kernel.

Selection of bandwidth:
The only parameter that can be changed is h, the bandwidth of the kernel, which determines how much influence the neighbouring points should have. It is very important that this is set correctly, otherwise there can be problems with both under- and oversmoothing. This will result in the population not being represented well as it can be seen in figure 3.12.



(a) Undersmoothed at $h = 0.1$

(b) Optimally smoothed at $h = 2.5$

(c) Oversmoothed at $h = 10$

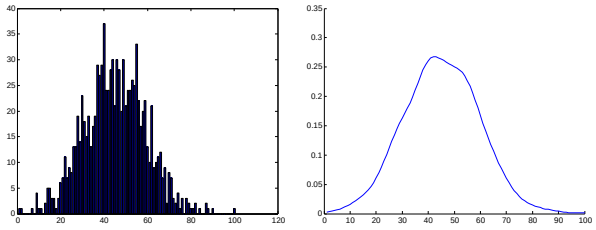
Figure 3.12: Example of Kernel density estimation with Gaussian kernel and the bandwidth h parameter.

There are methods for calculating the optimal bandwidth h for kernel density estimation with Gaussian kernel.

Silverman's rule of thumb [20]:
For univariate data distributions there is a simple, but effective way of obtaining the optimal bandwidth h for the kernel density estimator. By calculating the sample standard deviation $\hat{\sigma}$ it is possible to directly calculate h:

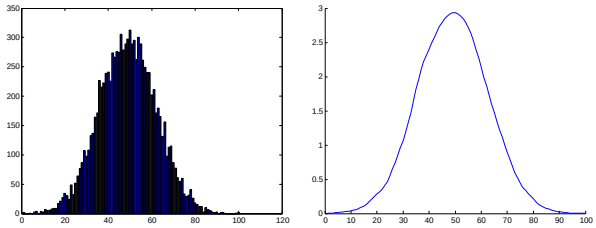
$$\hat{h} = 1.06\hat{\sigma}n^{-1/5} \tag{3.29}$$

As the following examples, figures 3.13 , 3.14 and 3.15, the estimation of h using equation 3.29 works well.



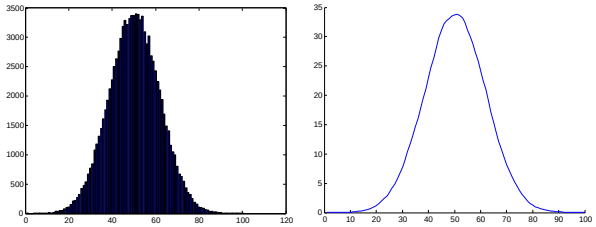
(a) Random generated normal distributed histogram. (b) Estimated distribution.

Figure 3.13: Estimated distribution with $N = 1000$, $\sigma = 14.29$ and $h = 3.8$.



(a) Random generated normal distributed histogram. (b) Estimated distribution.

Figure 3.14: Estimated distribution with $N = 10000$, $\sigma = 13.16$ and $h = 2.21$.

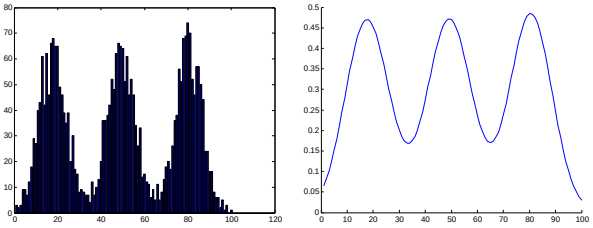


(a) Random generated normal distributed histogram. (b) Estimated distribution.

Figure 3.15: Estimated distribution with $N = 100000$, $\sigma = 11.67$ and $h = 1.23$.

As it can be seen from figures 3.13 , 3.14 and 3.15 the bandwidth becomes smaller when more samples are introduced. This makes perfect sense, since more samples will make the histogram be closer to the actual underlying distribution. Therefore less smoothing will be necessary.

The last examples given have been unimodal, but Silverman's rule of thumb can also be applied to multimodal distributions. This is crucial since distributions may very well be of this kind. An example of fitting a distribution to a trimodal histogram can be seen in figure 3.16, which is well represented.

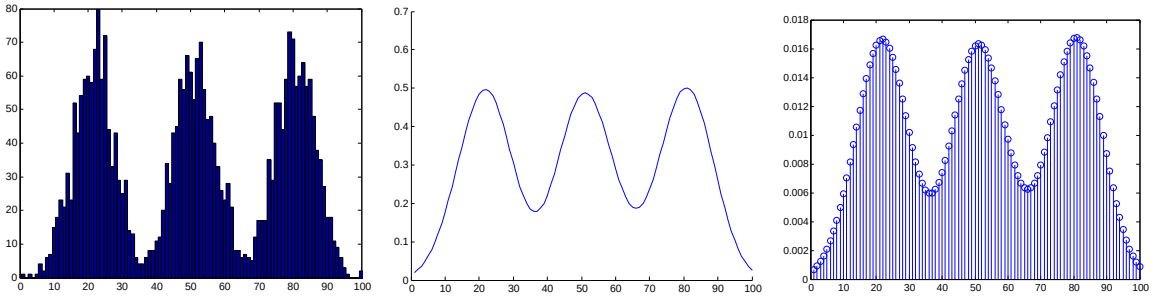


(a) Random generated trimodal normal distributed histogram.

(b) Estimated distribution.

Figure 3.16: Estimated distribution with $N = 3000$, $\sigma = 26.33$ and $h = 5.62$.

An example of using kernel density estimation combined with generating a pmf from a histogram can be seen in figure 3.17.



(a) Random generated trimodal normal distributed histogram.

(b) Estimated distribution

(c) Discretizised to normalized pmf.

Figure 3.17: Example of Kernel density estimation with Gaussian kernel and the bandwidth h parameter.

Chapter 4

Design and implementation

4.1 Introduction to design and implementation

In this part the design of the methods, features and classifiers will be explained. The design will extensively use the methods explained and discussed in the theory part.

The structure of the design for motion, size and texture features and classifiers are from the paper which this project is based on [45], but methods have been altered to improve results. The design for direction feature and classifier have been made entirely by the author.

An brief overview of the implementation will be given which should be sufficient to understand how the flow of the program is and how it is structured. The different blocks in figure 4.1 will be explained.

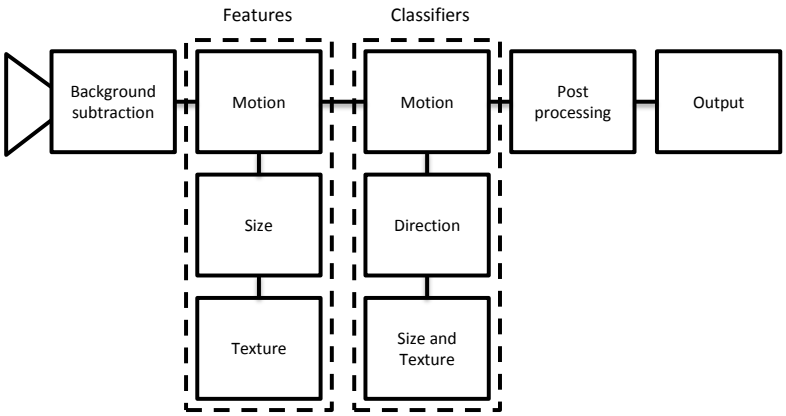


Figure 4.1: Overview of the system.

4.2 Design

4.2.1 Foreground segmentations

The foreground segmentation or background subtraction is done so that when features have to be extracted, it will only be from the foreground. This makes it possible to use the feature size, since this is modelled on the foreground pixels. It somewhat solves the aperture problem for motion feature when using standard Lucas-Kanade without ”good features to track”, since only

foreground pixels are tracked. For the texture feature it assures that only regions with foreground pixels are modelled and thereby lowers the modelling of the background.

The method using FGD [40] included in OpenCV has been chosen for the implementation since this has the lowest computational requirements while still achieving good results. In the paper which this method is based on, they use their own background subtraction algorithm [46]. Unfortunately this is not publicly available and therefore could not be used in this project.

4.2.2 Regions

The method works almost independently within regions, apart for some post processing. Every input frame is divided into equal size and quadratic regions where features and classifiers work independently for each region. A specific region is denoted as $R(i, j)$ and the number of regions depends on the size of the regions N. As illustrated in figure 4.2.

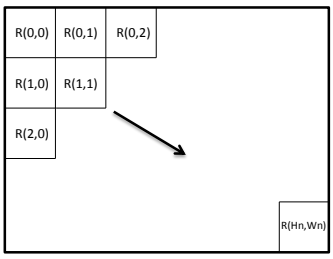


Figure 4.2: Separation of regions.

4.2.3 Feature: Motion

The motion feature in the paper which this project is baed on uses the Lucas-Kanade optical flow algorithm which computes the velocity vector for each pixel in both spatial directions from one frame to the next. The flow is calculated on each pixel, but only used for the foreground pixels and the final feature is a scalar that represents the average motion:

$$\hat{mot}_t(i,j) = \frac{1}{N_f} \sum_{n=0}^{N_f} \parallel [v_x^{(n)}, v_y^{(n)}]_1 \parallel \tag{4.1}$$

Where for each foreground pixel n, $v_x^{(n)}$ and $v_y^{(n)}$ is the optical flow in both spatial directions and N_f are the total number of foreground pixels.

To further reduce noise the motion scalar for region is averaged with the motion scalar for the same region in frame $t-1$ and frame $t+1$:

$$mot_t(i,j) = \frac{1}{3} \sum_{u=t-1}^{t+1} \hat{mot}_u(i,j) \tag{4.2}$$

The same approach will be used when testing the method with Horn-Schunck.

For Pyramid Lucas-Kanade the average motion will not be normalised within region by occupancy. Since this is a sparse method, the number of flow vectors will almost always be less than the

number of foreground pixels. Therefore the normalisation will be calculated with the number of flow vectors within the region, as in the following equation.

$$\hat{mot}_t(i,j) = \frac{1}{N_{fv}} \sum_{n=0}^{N_f} \parallel [v_x^{(n)}, v_y^{(n)}]_1 \parallel \tag{4.3}$$

Where for each optical flow vector n, $v_x^{(n)}$ and $v_y^{(n)}$ is the optical flow in both spatial directions and N_{fv} is the total number of flow vectors within the region.

4.2.4 Feature: Size

The feature size is based on the occupancy of the foreground pixels in each region combined with occupancy in the neighbouring regions. Neighbouring regions are used since the object might fill up more than one region. A Gaussian kernel is used to give more impact to the region that is calculated and less to the regions around it.

$$size_t(i,j) = \sum_{a=i-1}^{i+1} \sum_{b=j-1}^{j+1} G(a-i+1, b-j+1) o_t(a,b) \tag{4.4}$$

Where G is a 3x3 Gaussian kernel, and o_t is the occupancy of the region.

4.2.5 Classifier for motion and size

The motion and size classifiers are trained in an off-line training where the motion/size of each of the frame in the training set is found. This ends up with a histogram that is then smoothed, discretized and normalized to obtain a probability mass function. The theory behind creating the pmf is explained in section 3.5 .

The pmf will be used for testing whether or not a region in the test phase is abnormal or not, by a simple threshold. If the following equations are true, the region is abnormal:

$$\hat{p}_{mot}(mot_t(i,j)) < T_{motion} \tag{4.5}$$

$$\hat{p}_{size}(size_t(i,j)) < T_{size} \tag{4.6}$$

Where T is a decision threshold.

4.2.6 Texture

Feature: Texture

The texture feature is based on 2D Gabor wavelets in four directions: 0°, 45°, 90°and 135°. The feature descriptor is then the sum of the magnitudes of the wavelet in the different directions:

$$txt_t(i,j) = [m_0 \;\; m_{45} \;\; m_{90} \;\; m_{135}] \tag{4.7}$$

The texture vectors are only sampled for regions that have at least one foreground pixel to avoid modelling the background.

Classifier: Texture

Where the classifier for motion and size are not adaptive, the texture classifier is. For a 4-dimensional feature vector it is impractical to estimate the distribution. This would require many inputs to obtain a meaningful feature space and the computational costs would be high. Instead

the classifier is a codebook where the distance to the entries in the codebook to the input vector is a measurement of abnormality.

For the distance measurement Pearson's correlation coefficient is used. The reason for using this distance measure is that the Pearson's incorporate normalization of the texture contrast variations, by subtracting the mean. A standard euclidean distance will e.g. not do this. Furthermore the returned distance is between -1 and 1 which makes it suitable for comparison with a threshold.

To train the classifier the first 4D texture descriptor in each region is taken to be the first entry. Every input after this is then measured for its similarity using Pearson's correlation coefficient [17]:

$$p(a,b) = \frac{(a-\mu_a)\cdot(b-\mu_b)}{\parallel a-\mu_a \parallel \parallel b-\mu_b \parallel} \tag{4.8}$$

Where μ_x is the mean of vector x and $p(a,b)$ is in the interval $[-1,1]$.

If the input vector when compared to all entries in the codebook is more than a threshold then the codebook entry with the highest correlation coefficient is updated and the output of the classifier is normality. The update of a codebook entry is as follows:

$$c_k^{new} = c_k^{old} + \frac{1}{W_k+1}(x_{in} - c_k^{old}) \tag{4.9}$$

Where c_k is the best matching codebook entry, W_k is the number of vectors so far assigned to the codebook entry k and x_{in} is the input descriptor.

If the correlation coefficient to all codebook entries is less than a threshold, the input is added as an entry to the codebook and the output of the classifier is abnormality.

The threshold is set to 0.9 in the paper which this project is based on.

4.2.7 Cascade classifier

The classifiers work in a cascade system. First the classifier for motion is consulted and if it does not output an abnormality then the classifier for size and texture is consulted. The reason why size and texture both have to be abnormal is because size alone does not necessarily constitute an abnormality. A group of people standing close could have the appearance of an object being abnormal. By combining size and texture, the false positives generated by size should be eliminated by texture.

1. If motion classifier outputs an abnormality then the region is abnormal.
2. If the motion classifier outputs normal, consult size and texture classifier which should both output an abnormality for the region to be considered abnormal.

4.2.8 Feature: Direction

To determine the directions within a region, optical flow will be used. Where the gradients and their sum was used in calculating the motion, the two different motion vectors will be used in calculating the direction.

For each pixel that is in the foreground a direction between 0°and 359°should be calculated. To calculate the direction the equation in 4.10 is used. The reason for using atan2 instead of atan is that the returned range is $[-\pi;\pi]$ where atan returns $[-\pi/2;\pi/2]$. If the returned values are less than zero, 2π will be added and all the values will be in range $[0;2\pi]$.

$$\Theta = atan2(\frac{dy}{dt}, \frac{dx}{dt}) \tag{4.10}$$

The directions can then be used to calculate the general direction of a region. Where as with the motion it is possible to sum the gradient of each pixel, this is not the case with the direction. Therefore the approach for generating a model and testing if a region is abnormal, will have to be different from this.

A four dimensional histogram will be used as the direction feature. The binning will be done in the following ranges:

- Bin 1: $[45° : 135°[$
- Bin 2: $[135° : 225°[$
- Bin 3: $[225° : 315°[$
- Bin 4: $[315° : 360°]$ and $[0° : 45°[$

Figure 4.3 illustrates this.

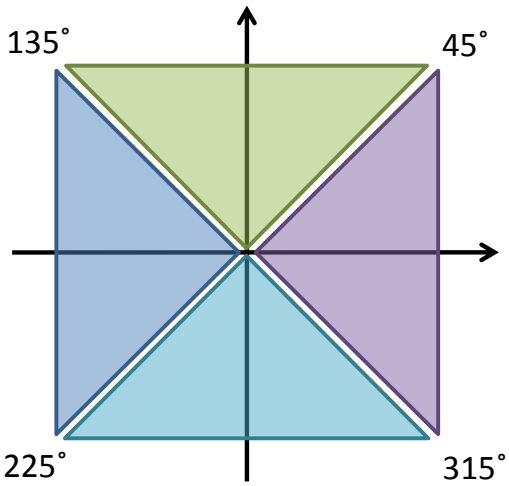


Figure 4.3: Figure illustrating the direction ranges.

4.2.9 Classifier: Direction

In the training phase a four dimensional vector will be contentiously updated for each region. The update consists of adding direction data for each frame. When the classifier is build the vector will then be normalized to the sum of one. To classify an incoming direction vector it is simply a matter of subtracting the two normalized vectors component-wise and summing the absolute values. This will be compared to a threshold which if surpassed will output an abnormality.

$$v_{direction} = \sum \left| \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} - \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \right| \tag{4.11}$$

Where h is the normalised classifier vector and x is the normalised incoming direction feature vector.

Testing whether or not a region in the test phase is abnormal or not, can be done by a simple threshold. The region is abnormal if:

$$v_{direction} > T_{direction} \tag{4.12}$$

Where T is a decision threshold.

4.2.10 Post-processing

To minimize noise a simple, but effective method, for post-processing the abnormality is done. It uses the neighbouring spatio-temporal regions to detect if the abnormality of a region is simply due to noise in the one frame, or if the region is abnormal in multiple frames, which indicates that it is not classified as abnormal due to noise. If a region is classified as abnormal then at least two neighbouring cells in frames $t-1$, t and $t+1$ must be classified as abnormal for the region to be abnormal. Otherwise it is re-classified as being normal. An illustration of the noise removing concept can be seen in figure 4.4.

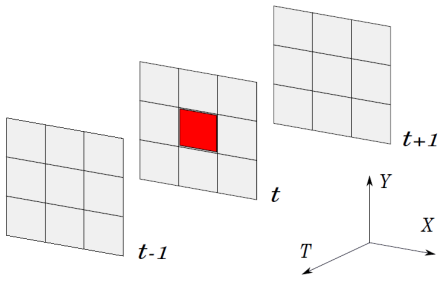


Figure 4.4: Spatio-temporal noise removal. Image is from the paper[45]

This project will use only one abnormal neighbouring region in each temporal frame for classifying a region as being abnormal.

4.3 Implementation

The method is implemented using C# with standard libraries and Emgu [7] which is a wrapper of OpenCV for C#. It includes most of the OpenCV functions and was chosen due to the simplicity of C# over less managed languages.

4.3.1 Class-diagram

For overview of the implementation a simplified UML diagram 4.5 was used. There are no fields in the diagram and not all methods are shown.

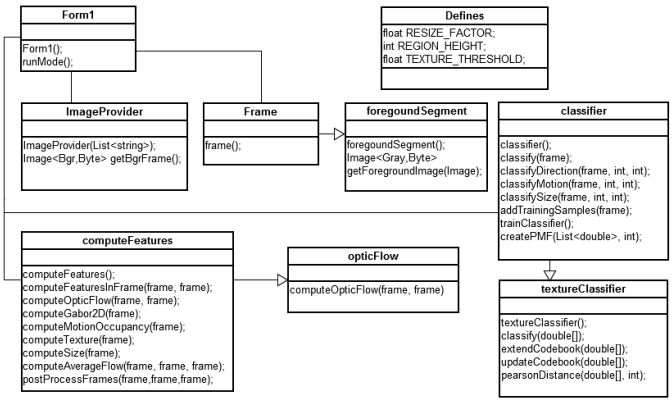


Figure 4.5: Simplified UML digram for implemented program.

4.3.2 Frame class

The frame class is a class that includes everything that is required for specific frame. A new instance of the frame class will be created each time a new frame is loaded from the input stream.

This newly created instance is then added to a list of frames which is used for processing the frames in the different steps of the algorithm. Once a frame is completely done being processed it is removed from the list to avoid allocating too much memory.

The frame class contains the following fields:

```
1         public struct regionInfoStruct {
2           public int width,height,posx,posy;
3         };
4
5         public struct regionDescriptorStruct {
6             public double motion, motionPreNormal, occupancy, size,
                  opticalFlowPoints;
7             public bool motionFeature, sizeFeature;
8             public bool abnormalPreMotion, abnormalPreSizeText,
                  abnormalPreDirection, abnormalPreSize;
9             public bool abnormalPostMotion, abnormalPostSizeText,
                  abnormalPostDirection;
10            public double[] texture;
11            public double[] direction;
12
13            //Variables for tests and read outs
14            public double[] closestTexture;
15            public double textureDistance;
16        };
17
18        public bool frameLevelAbnormalPost;
19
20        public static int imgWidth,imgHeight;
21        public static int yRegions, xRegions;
22        public static regionInfoStruct[,] regionInfo;
23        public static double[,] SizeGaussianKernel;
24        public Image<Bgr, Byte> colorImage;
25        public Image<Gray, Byte> bwImage, fgMask, flowImg, OpticalFlowFrame;
26        public Image<Gray, Single> flowx, flowy;
27
28        public List<Image<Gray, float>> textureImg;
29        public static foregoundSegment _ForgroundSegment;
30
31        public regionDescriptorStruct[,] region;
```

The 2D array *regionInfo* contains the static information about the region, which is the size. The 2D array *region* contains all the dynamic information about the region, the calculated features and the classifiers output.

Furthermore the class contains input image, processed images, a flag that states if the frame is normal or abnormal and the number of regions.

4.3.3 Timing diagram

There are several processes that are dependent on another process being done. For instance, it is not possible to compute the average flow before the flow has been computed and a frame cannot be post-processed before both the previous and post frame is completely classified.

Therefore it is very important to have an overview which makes sure that the different methods are executed at the right time. The diagram in figure 4.7 gives this overview.

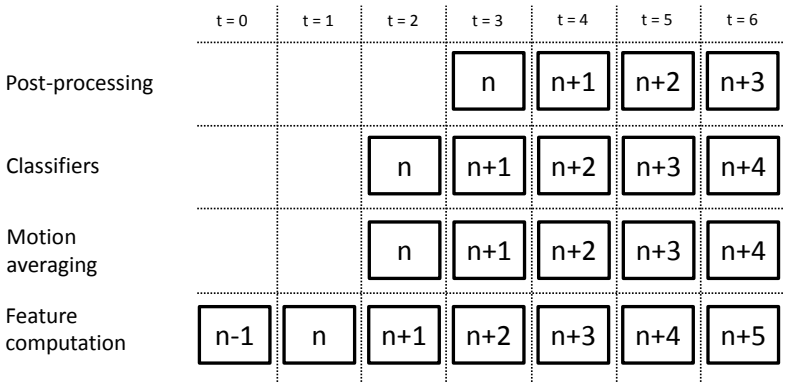


Figure 4.6: Timing diagram for overview of when a method can be executed.

Where the notation n defines the frame number in a series of frames.

This led to the following pseudo code:

```
1: while inputimage is not NULL do
2:   if frameList.Count > 1 then
3:     compute features for frame at frameList[frameList.Count - 1]
4:   end if
5:   if frameList.Count > 3 then
6:     compute average flow for frame at frameList[frameList.Count - 2]
7:   end if
8:   if frameList.Count > 3 then
9:     compute average flow for frame at frameList[frameList.Count - 2]
10:    classify for frame at frameList[frameList.Count - 2]
11:  end if
12:  if frameList.Count > 4 then
13:    post process for frame at frameList[frameList.Count - 3]
14:  end if
15: end while
```

The different methods have been put in a thread each to optimize performance when the CPU has multiple cores. Some effort was put into utilizing a GPU for pixel-wise processes. This was abandoned due to overhead in the memory allocation and thereby lack of measurable optimization.

4.3.4 Graphical user interface

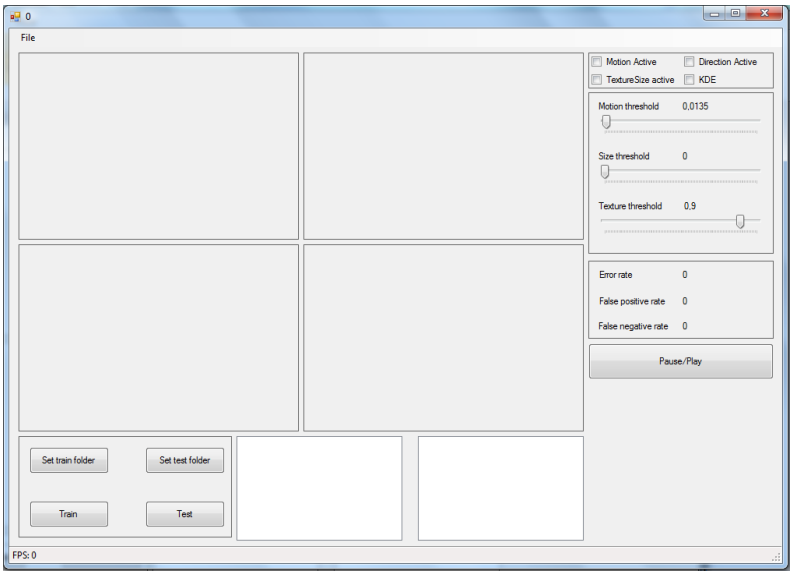


Figure 4.7: Graphical user interface.

The four squares show the input frame, the output from the background subtraction algorithm, the preprocessed frame with classification and the postprocessed frame with classification. By the use of checkbuttons, the features and classifiers can be selected to be active, the different thresholds can be selected by sliders and the error rate is updated on-the-fly if ground truth is included with the input frames. The stream can be paused and during the test phase, the variables in each region can be read out by placing the cursor on a specific region.

Chapter 5

Test and results

5.1 Introduction to test and results

In this chapter the methods proposed in the design part will be tested. This should reveal to what extend the methods perform and what the issues are. Test will be conducted on three datasets. The UCSD anomaly detection dataset [25] for testing the motion, size and texture features and classifiers. A synthetic dataset and a real world dataset for testing direction feature and classifier, created by the author.

After each test a small discussion will be conducted regarding the results.

Most of the parameters in the program are static through all tests, which makes the method testing more generic and does make sure that the tests are not suited to the optimal performance. A list of parameters can be seen in refsec:parameters.

5.1.1 Motion, size and texture tests:

For the UCSD dataset the following will be tested for the motion feature and classifier, at different motion thresholds:

- Optical flow algorithms.
- Region size.
- Number of histogram bins.
- The effect of kernel density estimation.

The size and texture feature and classifier will be tested at different thresholds with fixed Gabor filter parameters and fixed Gaussian kernel for size averaging.

Last an unsupervised training test and a pixel-level detection test will be conducted and the method will be compared to other methods using the same dataset.

5.1.2 Direction tests:

Following test will be conducted for the synthetic dataset:

- Optical flow algorithms.
- Incrementing method.

For the real-life dataset at different direction thresholds:

- Region size.
- Optical flow algorithms.

- Incrementing method.

A test of computational requirements will also be done and after each test there will be a small discussion concerning the results.

5.1.3 Tests measurement methods

The results will be measured using the following measurements:
False positive rate(FNR):

$$FPR = \frac{FP}{FP + TN} \tag{5.1}$$

Where FP is the number of false positive detections, FN is the number of false negative detections and TN is the number of true negative detection.

False positive rate(FPR):

$$FNR = \frac{FN}{FN + TP} \tag{5.2}$$

Where TP is the number of true positive detections.

Equal error rate (EER):
The equal error rate is where the false positives are equal to the false negatives and a measurement for performance that can easily be compared between methods.

Error rate (ER):

$$ER = \frac{FP + FN}{N} \tag{5.3}$$

Where N is the number of detections in total.

5.1.4 Requirements

The following points from the requirement specification 2.12 are listed and their involvement in the tests is explained:

- *Tuning the algorithm for a given video stream should be simple and fast — this process is done by technicians and for many dozens of cameras at every installation site*:

  As mentioned all parameters, except if specifically written are fixed during the tests. Thresholds for motion, size, texture and direction are the only parameters that should vary through every test. Parameters are listed with their default value in appendix 11.

- *The algorithm should be adaptive - the environment may change (different times of day for example)*:

  This is tested implicit with the background subtraction model.

- *Should be able to run in real-time*:

  A test of computational requirements will be made.

- *Robustness - many real-life scenes are crowded and cluttered*:

All real-life test data, in this project, is crowded and cluttered.

- *Should be able to detect more than one abnormality as those listed in section 2.7*:

  Test will be performed on motion, size, texture and direction.

- *Fully automatic learning (unsupervised) and operation, with a (desirably) short learning period*:

  A test will be conducted on both supervised and unsupervised training approaches.

- *Low computational requirements - either the algorithm will run on a DSP card installed with the camera ("smart camera") or on a small number of PCs handling multiple cameras. It is unreasonable to require a PC for every camera*:

  A test of computational requirements will be made and evaluated.

- *Predictable performance - which should be adequate in many operationally interesting scenes*:

  The tests will show if the output is predictable.

5.2 Motion, size and texture test

5.2.1 Dataset for testing motion, size and texture

The UCSD dataset is created for abnormality detection algorithms and will be used for testing motion, size and texture features and classifiers. The dataset consists of two subsets, Ped1 and Ped2, which both have training and testing parts. An introduction to the dataset can be seen in appendix 8.

Even though the UCSD dataset includes both abnormalities that should be detected by motion and by size/texture, the dataset will be tested upon individually for each classifier. This will make it easier to determine which methods within the classifiers works best.

The tests will be done on different parameters, but the evaluation will always be on the EER.

For the motion test only the Ped1 part of the dataset will be used and the size and texture only the Ped2 part of the dataset will be used.

The classifier will, in both cases, be trained with all the training samples and tested with all the testing samples. If a frame in the test has one or more abnormality it will be deemed as abnormal. This will after the test be compared with the ground truth.

(a) Training frame. (b) Test frame with abnormality: golf car.

Figure 5.1: Ped1: A sample training frame and one test frames.

(a) Test frame with abnormality: wheelchair.

(b) Test frame with abnormality: bicycle.

Figure 5.2: Ped1: Twotest frames.

5.2.2 Motion tests

To test the motion feature and classifier, several parameters will be changed and results of these will be evaluated.

The parameters that will be tested are:

- Optical Flow algorithm: Standard Lucas-Kanade, Pyramid Lucas-Kanade and Horn–Schunck. The algorithms are described in 3.3 and should yield different results.
- Region size: The size of each region.
- Histogram bins: Number of histogram bins.
- Kernel density estimation: If using kernel density estimation will have an positive effect on the result.

Optical flow algorithms:

The different optical flow algorithms have been tested with different probability thresholds which yields the curve in figure 5.3. This shows a comparable result of the three different algorithms.

The search window size for the optical flow has been set to 15 x 15 pixels in the Lucas-Kanade based methods, with three pyramid levels and 1000 ”good features to track” used in the Pyramid Lucas-Kanade. For the Horn–Schunck method the stop iteration criteria has been set to 20.

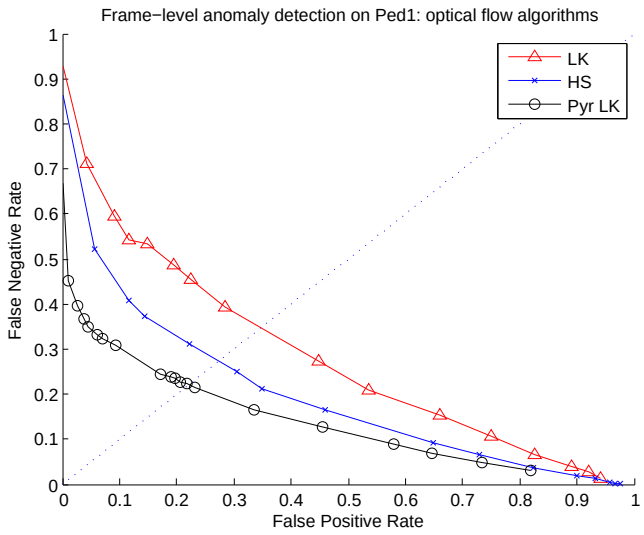


Figure 5.3: Curve for Pyramid Lucas-Kanade, Lucas-Kanade and Horn-Schunck at 20 histogram bins, no kernel density estimation and regionsize of 16 x 16.

Discussion of results:
As it can be seen the Pyramid Lucas-Kanade outperforms the other two methods at EER and elsewhere. This is no doubt because it utilises the ”good features to track” which avoids calculating optical flow where the aperture problem is present.

Region size:

Region size of 8x8, 16x16 and 32x32 have been tested with all other parameters being the same and yield the curve in figure 5.19.

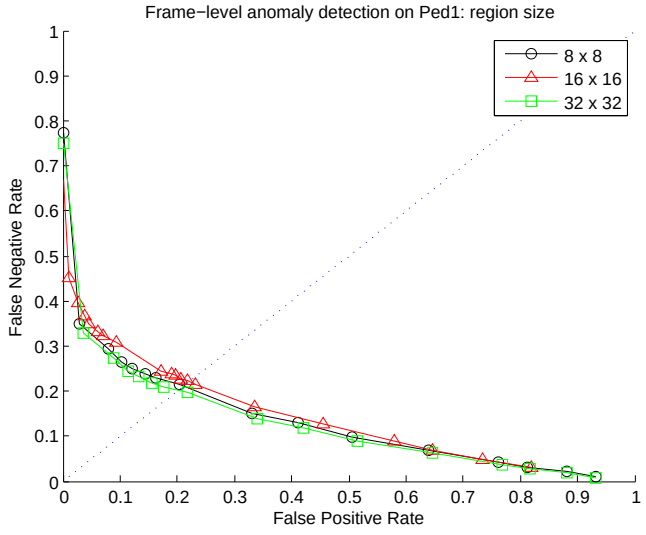


Figure 5.4: Curve for Pyramid Lucas-Kanade at 20 histogram bins, no kernel density estimation and regionsize of 8 x 8 px, 16 x 16 px and 32 x 32 px.

Discussion of results:
The three regions sizes generate very similar results. This is very fortunate since it makes the system more robust and predictable, as compared to if the region size had a big impact on the results. In the paper which this project is based on they state that 16 x 16 px is the best option, based on their preliminary results. This might be because of the size and texture feature and classifier, but there seems to be no reason for not going with 8 x 8 or 32 x 32 for motion classification.

Histogram bins:

The effects of using 5, 10, 20 or 30 histogram bins have been tested with all other parameters being the same and yield the curve in figure 5.5.

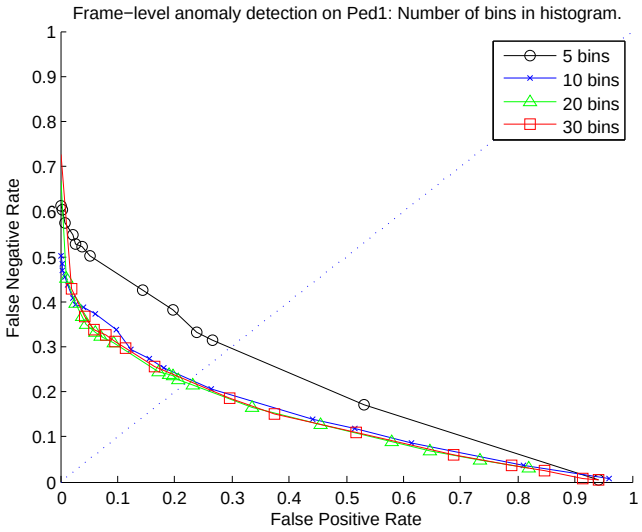


Figure 5.5: Curve for Pyramid Lucas-Kanade at 5, 10, 20 and 30 histogram bins, no kernel density estimation and regionsize of 16 x 16.

Discussion of results:
There is little difference between using 10, 20 or 30 bins histogram. For bigger histogram sizes this should be the same, since more bins should produce same results at other lower probability thresholds when the histogram is normalised to one, in creation of the pmf.

As for the 5 bins test there is not doubt that there is a lower amount of histogram bins that can be used, while achieving the same performance. This is because the underlying distribution of motion would disappear when creating the pmf.

Kernel density estimation:

To see if there is an effect by using kernel density estimation the exact same tests have been conducted with and without kernel density estimation. The results can be seen in figure 5.11.

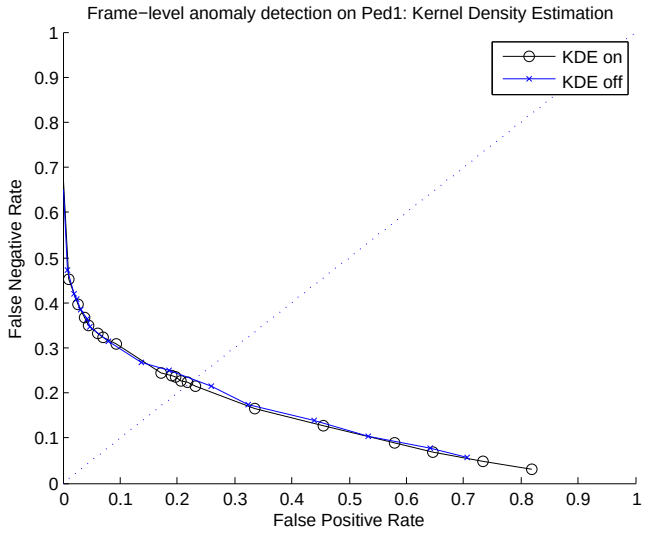


Figure 5.6: Curve for Pyramid Lucas-Kanade at 20 histogram bins with and without kernel density estimation and regionsize of 16 x 16.

Discussion of results:
The use of kernel density estimation changes nothing on the performance rate. This indicates that there is no problem with coarse binning and that there is no reason for kernel density estimation.

5.2.3 Texture and Size tests

To test the size and texture classifier the important parameters will be changed and results of these will be evaluated.

By visually inspecting the output from the convolution of the input image with the Gabor filters, it is clear what parameters give the best results. The kernel size was set to approximately half of the standard region size, namely 7 x 7, and then slightly less to get symmetric responses on each side of the pixels. The radial frequency ,ω, was set to 2.3 since this value seemed to produce the best results in differentiating objects. In figure 8.2 the results of different values of ω is investigated.

(a) Image convoluted with Gabor filter with too low frequency $\omega = 0.9$

(b) Image convoluted with Gabor filter with optimal frequency $\omega = 2.3$

(c) Image convoluted with Gabor filter with too high frequency $\omega = 3.6$

Figure 5.7: Ped2: Test004 - image 81 at 0°with different radial frequencies ω.

Furthermore, to limit the parameters that can be used in the test, the texture threshold is set to 0.9 as in the paper [45].

There will be no test of kernel density estimation and number of histogram bins, since it has already been established in the motion test that there is no effect of the former and the latter is well proportioned at 20 bins.

For the Gaussian kernel to average the size (occupancy) of each region, it has been chosen to use a standard 3 x 3 Gaussian Kernel with mean $\mu = 0$ and variance $\sigma^2 = 0.6$. The kernel can be seen in figure 5.1. It is important to choose a kernel that with high enough variance to take the neighbouring occupancies into consideration, but also not too big which would render the surrounding bins significantly more important.

0.0277	0.1110	0.0277
0.1110	0.4452	0.1110
0.0277	0.1110	0.0277

Table 5.1: Gaussian kernel for averaging size of region.

The only parameter that will be changed is the threshold T_{size} which regulates when a size is found to be abnormal or not. The subset of the UCSD dataset, Ped2 will be used for testing the size and texture classifier. The results can be seen in figure 5.8

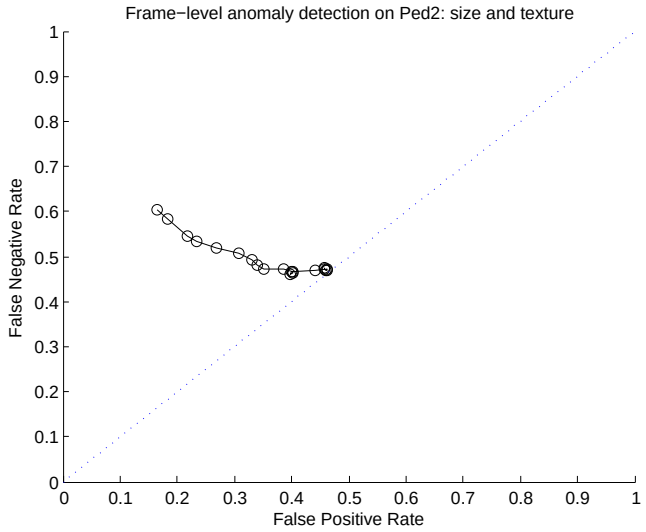


Figure 5.8: Curve size texture classifier with a kernel size of 7x7, a radial frequency ω of 2.3 and region sizes of 16 x 16.

Discussion of results:
It is clear that the results are not nearly as good as with the motion feature, but still better than a classifier with random output. This is partly due to there being more motion abnormalities than size and texture in the datasets. Even though there are problems with the size and texture feature and classifier.

The feature does not seem to make good enough differentiating of e.g. a pedestrian and a car. There are definitely a visual and measurable difference especially in the magnitude of the output. This however this is often lost in the classification process because of the use of Pearson's correlation coefficient as a measurement. Since the input is normalised the overall magnitude is of no importance, which in theory is good since this will make it invariant towards how much the objects stand out from the background. By selecting a non normalising distance measurement the classification would become invariant towards this and the method might produce many errors as well.

Further research and better features will have to be found to make the size and texture methods viable to use in automated surveillance.

5.2.4 Post processing test

To test the effects of the post processing, the method's parameters will be changed and the results will be compared at EER. As explained in section 4.2.10 the authors of the paper propose that a region is only abnormal if there is also two abnormal neighbouring regions in the current, post and previous frame.

The following test scenarios will be tried:

$t-1$	t	$t+1$
2	2	2
1	2	1
2	1	2
1	0	1
0	0	0

Table 5.2: Post processing tests

Where t is the current frame and the numbers indicate how many of the neighbouring regions that have to be classified as abnormal, for the current region to be abnormal.

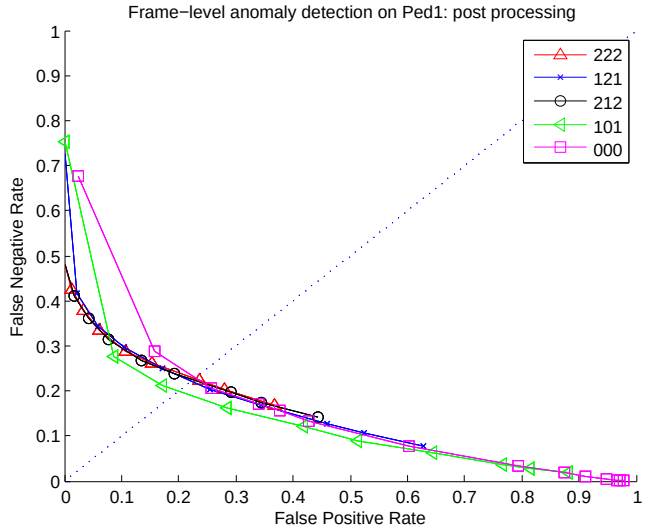


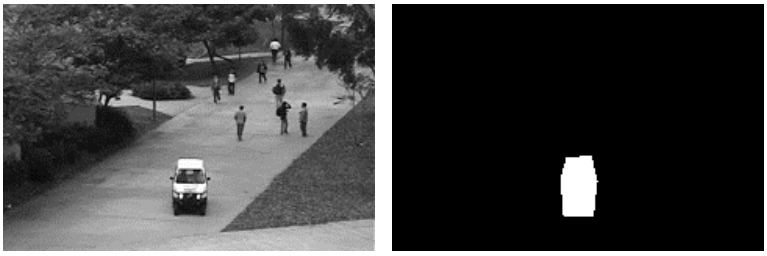
Figure 5.9: Curve for frame-level detection test on Ped1 with different post processing methods. Pyramid Lucas-Kanade at 20 histogram bins, no kernel density estimation and regionsize of 16 x 16.

Discussion of results:
At EER the post-processing method requiring an abnormal region to have an abnormal temporal neighbour in previous and post frame is outperforming the other methods. This is because it sorts out the falsely detected abnormal regions, but still does not remove too many of the truly detected abnormal regions.

There seems to be no difference between the other methods. The more neighbouring regions required for a region to be classified as abnormal, the higher the motion threshold T_{motion} has to be. As well as the fewer neighbouring regions required the lower the the motion threshold has to be.

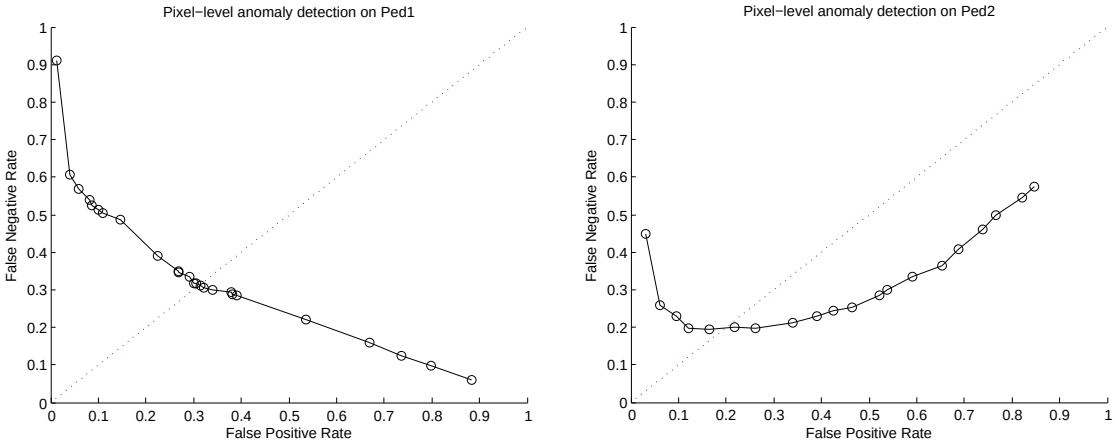
5.2.5 Pixel-level test

There are officially 10 Ped1 and 9 Ped2 sequences with ground truth pixel-level bitmap masks. However one of the sequences from Ped2 does not have matching ground truth bitmaps and it is therefore discarded. These masks can be used to evaluate the performance of the methods when the goal is to locate the abnormality precisely. As with the other papers using this dataset, the frame will be counted as correctly detected if at least 40 % of the pixels in the detected regions match the ground truth pixels. If not, the the frame will be counted as falsely detected.



(a) Frame from Ped1 (b) Corresponding ground truth mask

Figure 5.10: Frame 120 from Ped1 sequence 19.



(a) Ped1 pixel-level test (b) Ped2 pixel-level test

Figure 5.11: Curve for pixel-level detection test on Ped1 and Ped2, with Pyramid Lucas-Kanade at 20 histogram bins, no kernel density estimation and regionsize of 16 x 16.

Discussion of results:
The result for Ped1 at EER is 31% error rate which is approximately 10% higher than at frame-level detection for the Ped1 dataset. For the Ped2 dataset the EER is approximately 6% higher at 21%. This is very reasonable with the added constraints that are from the frame-level to pixel-level test and shows that the method works well for detecting the actual abnormalities.

5.2.6 Unsupervised training test

The former tests are with supervised training since this was the approach used by the other papers[45][44][43][38] that use the UCSD dataset for testing. This made comparison between the methods possible.

The unsupervised training test will be conducted, using only the motion feature, in the same fashion as the previous tests, but with the fixed parameters found to generate the best results in section 5.2.2. Optical flow method is Pyramid Lucas-Kanade, region size of 16 x 16, 20 histogram bins, no kernel density estimation and a motion threshold which is at EER.

To make the training data unsupervised, extra frames from the test data will be added to the training data that is used for creating the classifiers. This should emulate unsupervised training since the data now used for training will contain both abnormalities and normalities, as in the real world.

It should be taken into consideration that when a frame is normal, there might be much activity that represents this. When a frame is abnormal there might only be one region that is abnormal. This makes the overall normality greater even if there are as many abnormal as normal frames used for training the classifier.

The percentage of frames that are abnormal and used for the training are calculated by the following equation:

$$R = \frac{N_{abnormal}}{N_{normal} + N_{abnormal}} \tag{5.4}$$

Where N is the number of frames, $N_{abnormal}$ is the number of abnormal frames (noted in ground truth) and N_{normal} is the number of normal frames.

Not all the frames in the test data are abnormal. Therefore a frame is only abnormal if it is in the test data, but also flagged as abnormal in the ground truth.

The error rate is measure as:

$$ER = \frac{N_{FP} + N_{FN}}{N_{total}} \tag{5.5}$$

Where N is the number of frames, N_{FP} is the number of false positive frames and N_{FN} is the number of false negative frames.

Results

The results of the test can be seen in figure 5.12 with error rate, false positive rate and false negative rates.

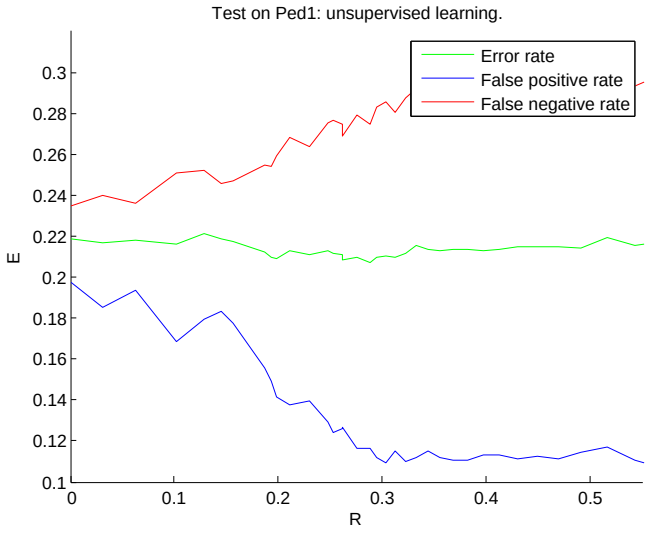


Figure 5.12: Test of learning style - with error rates and ratio of normal to abnormal frames.

To test if there is any significance, at their respective EER ,between training the classifier with specific training data or training the classifier with the test data the curve in figure 5.13 was created.

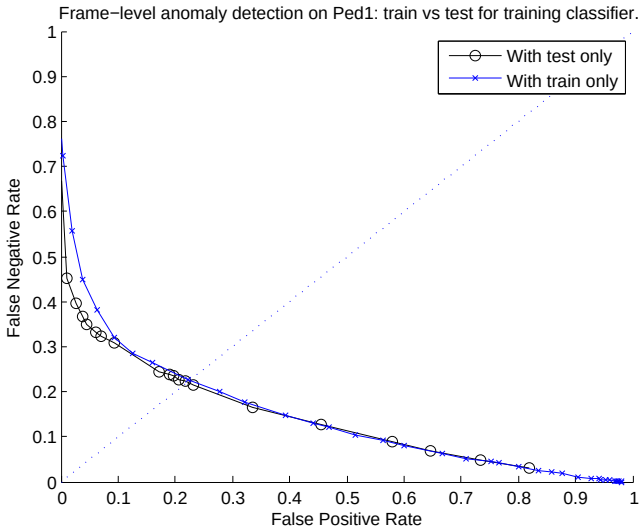


Figure 5.13: Test of learning style - using only training frames or only test frames for training the classifier.

Discussion of results:
It makes no significant difference on the error rate which data is used for training the classifier. This is due to the previously mentioned notion that a frame which contains an abnormality also contains many normalities. Since the classifier is normalized the somewhat sparse amount of abnormalities in the training phase has no major impact.

For the false positive and false negative rates there is more impact by the percentage of abnormal frames used. The more abnormal frames that are used in the training phase, the more bias the classifier will be towards recognising an abnormality as a normality. The false positive rate drops because of the same bias. The normalities, previously falsely detected as abnormal are now less than the motion thresholds.

At EER there is no seeming difference between using either the training data or test data for creating the classifier. The motion threshold is significantly higher at EER for the test conducted using the test data for training the classifier with T_{motion} at 0.06, compared to the standard approach with T_{motion} at 0.0375. The reason for the higher threshold can be seen in figure 5.12 with the inclining false negative rate and declining false positive rate.

The two tests indicates that the method is very suitable for an unsupervised training approach.

5.2.7 Cause of errors:

Since the size and texture feature and classifier are not working optimally there will of course be some of these abnormalities that are not detected. These abnormalities have the same motion as the pedestrians and are therefore not picked up by the motion classifier. This is what can be seen in figure 5.14 with the wheelchair in the lower right corner and the cyclists in the middle.

(a) Ped1: not detected abnormality. (b) Ped2: not detected abnormality.

Figure 5.14: False negatives on Ped1 and Ped2.

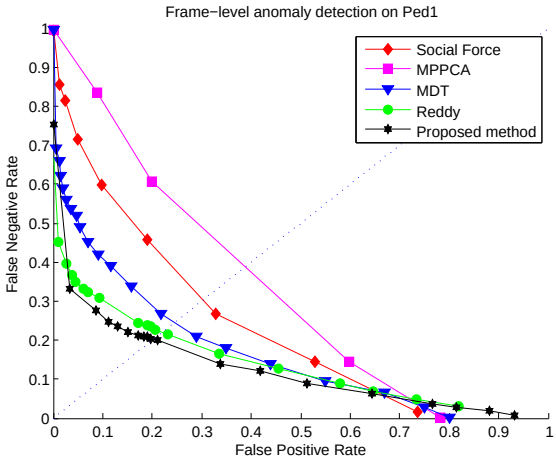
Many of the false positives are caused by the lack of motion in the region during the training phase. If no motion have been modelled in a region in the training phase, then motion in the test phase will output an abnormality. This is what can be seen in figure 5.15 by the pedestrian in the lower right corner and the pedestrian in the lower left corner. It could be discussed whether or not the ground truth annotations are completely correct, since it could be regarded as an abnormality when a region that is normally empty introduces motion.

(a) Ped1: falsely detected abnormality. (b) Ped2: falsely detected abnormality.

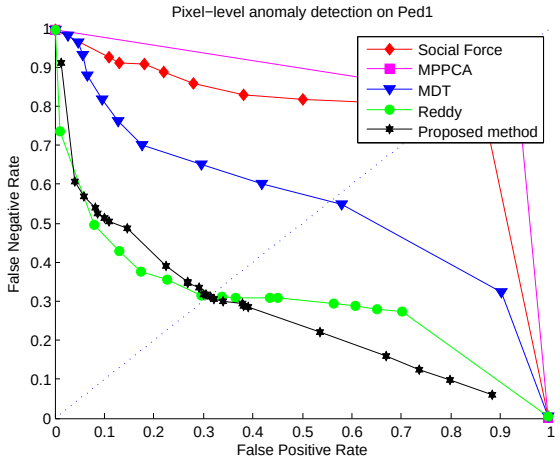
Figure 5.15: False positives on Ped1 and Ped2.

5.2.8 Comparison with other methods

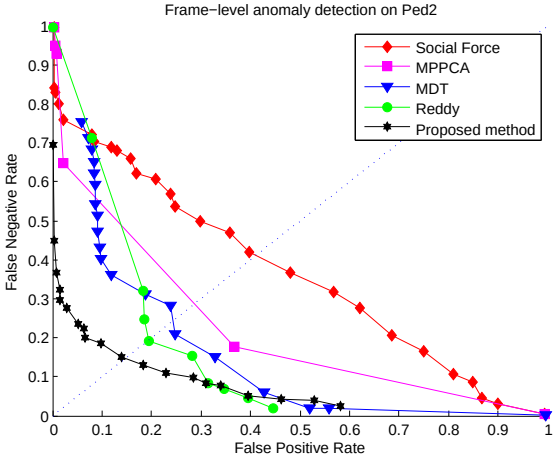
Since the size and texture classifier does not produce results that would improve the overall detection, this part have been no been used. Only the motion feature and classifier is used and this will be compared to the other methods using the UCSD dataset, which are: ”Reddy” [45] , ”Social Force” [44] , ”MDT” [43] and ”MPPCA” [38].



(a) Ped1 frame level detection.



(b) Ped1 pixel level detection



(c) Ped2 frame level detection

Figure 5.16: Comparison on the UCSD anomaly dataset with other methods.

Approach	Social Force	MPPCA	MDT	Reddy	Proposed method
Ped1	31.0%	40.0%	25.0%	22.5%	**20.0%**
Ped2	42.0%	30.0%	25.0%	20.0%	**15.0%**
Average	37.0%	35.0%	25.0%	21.2%	**17.5%**

Table 5.3: Equal error rates (EERs) for frame-level abnormality detection, obtained on the Ped1 and Ped2 subsets on the UCSD dataset

Approach	Social Force	MPPCA	MDT	Reddy	Proposed method
Ped1	79.0%	82.0%	55.0%	32.0%	**31.0%**
Ped2	-	-	-	-	**21.0%**

Table 5.4: EERs for the pixel-level abnormality detection.

Discussion of results:
Even without the size and texture classifier the method using Pyramid Lucas-Kanade outperforms the method which this project is based on [45] and others. There might not be a big improvements in percentage points to gain if a size and texture classifier worked perfectly, but it should be at least 3-4%. By that there should be much to gain from using the better optical flow algorithm Pyramid Lucas-Kanade, which is equally as fast which is explained in section 5.4.

For the pixel-level abnormality detection the proposed method outperforms the current methods and even the closest method which uses a presumably working size and texture feature and classifier.

5.3 Direction tests

5.3.1 Dataset for testing direction

The datasets for direction feature and classifier are created by the author. They consists of a synthetic dataset and a real-life dataset and are introduced in appendix 9.

(a) Frame from synthetic dataset. (b) Frame from real-life dataset.

Figure 5.17: Direction dataset: A frame from the synthetic and real-life dataset.

5.3.2 Direction test on synthetic dataset

The synthetic videos will be used for ground truth data when testing the concept and the different possibilities that lies within this. When computing the four dimensional vector for direction there is the possibility of either incrementing the direction bin by one, or adding the gradient to the direction bin. The hypothesis is that small gradients might have a higher error rate towards correct direction than larger gradients. This will be known after performing the tests. The optical flow computation method will be tested with all of the optical flow methods.

The different test that will be performed is:

1. Adding the gradient of the pixel to the direction bin.
2. Incrementing the direction bin.

Even though the four different videos basically are just mirrored there might be different results due to the background subtraction algorithm, but this should be negligible. Also, since the Pyramid Lucas-Kanade method uses the ”good features to track” there might be regions where a good feature is not present, even if it is a part of the foreground. These regions will produce a four dimensional vector with all entries being zero. These have been ignored when calculating the percentages. Still this test should provide solid ground for choosing method and algorithm for the direction based abnormality detection.

Results of direction tests

Test 1: First test is when incrementing the calculated direction bin in the direction vector with one for each pixel found as being foreground. The combined sum of each bin after the 300 frames and the amount of times each bin has been the biggest can be seen in tables 5.5 , 5.6 and 5.7.

	% Correct bin	% Correct max bin
Up	88.7%	98.3%
Down	88.9%	98.2%
Right	88.2%	98.6%
Left	87.5%	98.43%

Table 5.5: Results of Lucas-Kanade test of direction when incrementing bin with one.

	% Correct bin	% Correct max bin
Up	100%	100%
Down	100%	100%
Right	100%	100%
Left	100%	100%

Table 5.6: Results of Pyramid Lucas-Kanade test of direction when incrementing bin with one.

	% Correct bin	% Correct max bin
Up	67.4%	98.4%
Down	67.6%	98.7%
Right	71%	94.7%
Left	70.3%	93.4%

Table 5.7: Results of Horn-Schunck test of direction when incrementing bin with one.

Test 2: The second test is when adding the motion gradient to the calculated direction bin in the direction vector. Results can be seen in tables 5.8 , 5.9 and 5.10.

	% Correct bin	% Correct max bin
Up	65.5%	85.5%
Down	63.9%	88.7%
Right	41.2%	85.2%
Left	44.7%	85.2%

Table 5.8: Results of Lucas-Kanade test of direction when adding motion gradient to bin.

	% Correct bin	% Correct max bin
Up	100%	100%
Down	100%	100%
Right	100%	100%
Left	100%	100%

Table 5.9: Results of Pyramid Lucas-Kanade test of direction when adding motion gradient to bin.

	% Correct bin	% Correct max bin
Up	80.1%	99.2%
Down	75.5%	99.3%
Right	86.87%	95.1%
Left	69.1%	94.5%

Table 5.10: Results of Horn-Schunck test of direction when adding motion gradient to bin.

Discussion of results:
It is clear from the tables that the Pyramid Lucas-Kanade method works much better than the other methods. This is probably due to the lack of texture in the data which makes the aperture problem quite big for the standard Lucas-Kanade and Horn-Schunck, but not for the Pyramid Lucas-Kanade method since it utilises the ”good features to track”. Further testing will be done on real surveillance data in the next section 5.3.3

5.3.3 Test on real world dataset

The parameters that will be tested are:

- Optical Flow algorithm: Standard Lucas-Kanade, Pyramid Lucas-Kanade and Horn–Schunck.
- Region size: The size of each region.
- Incrementing method: If the motion gradient is added to the vector entry or if the vector entry is incremented by one.

The three different parameters will be tested in a range of different thresholds for classifying if the region is abnormal or not.

Optical flow algorithms:

The different optical flow algorithms have been tested with different thresholds which yields the curve in figure 5.18. This will show a comparable result of the three different algorithms.

The search windows size for the optical flow has been set to 15 x 15 pixels in the Lucas-Kanade based methods, with three pyramid levels and 1000 ”good features to track” used in the Pyramid Lucas-Kanade. For the Horn–Schunck method the stop criteria has been set to 20.

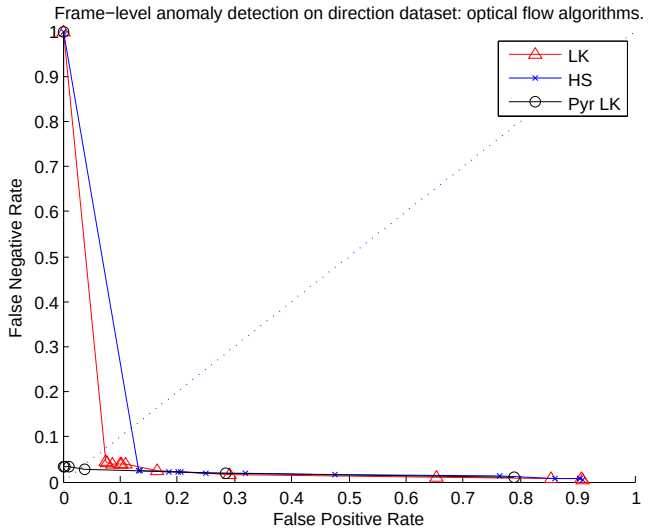


Figure 5.18: Curve for Pyramid Lucas-Kanade, Lucas-Kanade and Horn-Schunck at 20 histogram bins, no kernel density estimation and regionsize of 16 x 16.

Discussion of results: As it can be seen the Pyramid Lucas-Kanade outperforms the other two methods at EER. This is as in previous optical flow method test, no doubt because it utilises the ”good features to track” which avoids calculating optical flow where the aperture problem is present.

Region size:

For the regionsize the Pyramid Lucas-Kanade will be used since this showed the best EER in the former test. The same parameters will be set and the region size will be 8 x 8, 16 x 16, 30 x 30 and 45 x 45 pixels. The results can be seen on the curve in figure 5.19.

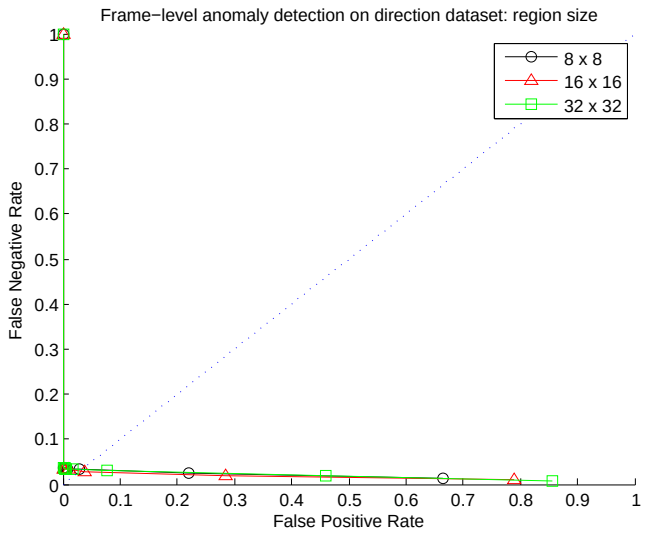


Figure 5.19: Curve for Pyramid Lucas-Kanade at 20 histogram bins, no kernel density estimation and regionsize of 8 x 8 px, 16 x 16 px and 32 x 32 px.

Discussion of results:
The three regions sizes generate very similar results and as previously mentioned this is very fortunate since it makes the system more robust and predictable compared to if the region size had a big impact on the results.

Iterating method:

To determine which of the two incrementing methods that works best, test have been conducted using both methods. The results can be seen in figure 5.20.

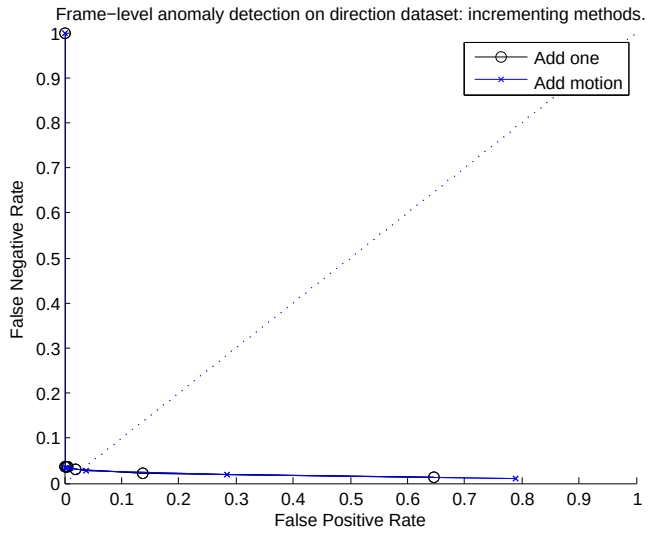


Figure 5.20: Curve for Pyramid Lucas-Kanade with regionsize of 16x16, with the two different incrementing methods.

The difference between the single increment and the gradient increment of the motion seems to be negligible, which again can be contributed to the Pyramid Lucas-Kanade using only points that track well between frames and produce excellent results in determine the correct direction.

5.3.4 Cause of errors:

Many of the false negative errors are caused by the lack of well tracked objects in the scene. As in the frames from 7714 to 7716 the white car in the lower left corner is not being tracked. This is because the ”good features to track” algorithm has not selected a single feature in this area, due to there not being any highly textured areas. Since the method works with a post-processing the frames before and after will also be affected which is why 7714 and 7716 is not recognised as abnormal. The foreground, optical flow and output of frame 7715 can be seen in figure 5.21.

(a) Output (b) Foreground (c) Flow

Figure 5.21: Frame 7715: Reason for false negative.

The false negatives are only a few frames in some of the reverse sequences, that happens in the one hour test video. Every car that runs in reverse is spotted at least once. Especially when they are close to the camera, since this gives the Pyramid Lucas-Kanade more pixels to track.

The false positives often occur where pixels are tracked badly. In frame 8054 this is the case - two pixels on a white car is selected by ”good features to track” and is found by the Pyramid Lucas-Kanade to move in different directions than they actually do. The foreground, optical flow and output of frame 8054 can be seen in figure 5.22.

(a) Output (b) Foreground (c) Flow

Figure 5.22: Frame 8054: Reason for false positive.

5.3.5 Results at optimal settings:

The optimal settings for this video is using the Pyramid Lucas-Kanade is using any region size. This produces approximately 30 false negatives and 40 false positives and is a total error rate of 0.30% which is deemed acceptable. It is also possible to tweak the threshold to make the algorithm produce less false positives if they are deemed more costly. It all depends on the application of the system.

5.4 Computational requirements

All tests are conducted on ASUS U46S with an Intel Core i5-2410M CPU running at 2.30 GHz and are an average FPS for an entire test sequence run.

	Ped1	Ped2	Direction
Size	238x158	360x240	320x240
Motion FPS	50	40	-
Size/Texture FPS	20	15	
Motion & Size/Texture FPS	18	12	-
Direction FPS			50

Table 5.11: Datasets with their respective FPS using different features.

Discussion of results:
With the motion, size and texture feature and classifier being active the method proposed achieves a better frames pr second than the method this project is based on [45] with an improvement of 50% with nearly the same hardware architecture. This is probably because the blocks of the different methods are threaded and runs in parallel.

A simple system could utilize only the motion feature and still achieve good results where five cameras could run at 10 frames per second each. This is important since one of the requirements was to not have one computer for each camera.

Chapter 6

Conclusion

The problem statement was: ***Is it possible to robustly and automatically detect abnormal events in video surveillance in real-time where the abnormal events are speed, direction and out-of-place objects?***

The tests show that it is definitely possible to make a system that can detect abnormal events in video surveillance where the abnormality is motion. A good detection rate has been shown on the UCSD anomaly detection dataset especially compared to other methods testing on the same dataset. Different optical flow methods have been tried and there is clearly an improvement by using Pyramid Lucas-Kanade with "good features to track". By avoiding the negative aspects of optical flow, namely the aperture problem, and only computing flow in areas where it is feasible, better error rates can be achieved.

For the size and texture abnormalities the tests showed that there was a small improvement over a completely random classifier. Although keeping the false positives to a minimum and combining this with the motion feature the error rates were higher than just using the motion feature. This is due to the feature descriptor not differentiating enough between pedestrians, cars, bicycles, wheelchairs etc. With a better feature descriptor the classifier should work.

Two datasets were created for detecting direction based abnormalities with the real-world dataset showing an error rate of 0.3%, which is very acceptable. The tests again showed that the best performance is achieved by using Pyramid Lucas-Kanade and the "good features to track". This method could be used for detecting ghost riders on highways or cars going the wrong way on a one way street. By minimizing the false positives, and thereby getting a few more false negatives, it should be possible to use this in actual employed surveillance systems.

The system could be used unsupervised. The unsupervised test on the UCSD anomaly dataset show that there is no difference between training the system with the training or test part of the dataset. This clearly indicates that as long as the detected activities are rare while training the classifier, they will be detected in the test or running phase of the program. Therefore it should be possible to simply install the system without having to input frames where no abnormalities are present.

For the robustness requirement the tests shows that region size, histogram bins, kernel density estimation and incrementing methods do not affect the results of the method. This shows that very few parameters will affect the performance rate of the system and therefore robustness is achieved.

The system is able to work in more than real-time. With only using the direction or motion it is be possible to connect four or five cameras to a single computer. With further optimization even more cameras could be connected.

A system have been created with achieves good results on motion based abnormalities and very

good results on direction based abnormalities with a high degree of robustness. It is able to run in real-time and apart from setting the thresholds for the classifiers it operates automatically. Further work has to be done on the size and texture classifier which would make the system even more generic towards detecting abnormalities in video surveillance.

The method proposed in this project outperforms the other current methods which operates on the UCSD dataset, mostly by evaluating and choosing the optimal optical flow algorithm and also includes a very good method for direction based abnormalities.

Chapter 7

Future Work

To test the genericness of the system more datasets should be tried where abnormalities are different. For instance for the direction classifier it could be interesting to test on two highways passing over each other and evaluate if the system can take the overlapping into account and still show good performance.

Further work has to be done on the texture feature to find a descriptor that better differentiates between objects. Tests with GLCM (gray level co-ocurrence matrix), SIFT, HOG or other descriptors could be tried as well as doing more test with different classifiers. A working texture classifier could be the extra that completes the project.

In the Ped1 part of the UCSD dataset and the real-life direction dataset there are perspective changes in the scene. Improvement of the detection rate might be found by using different region sizes in these scenarios. The region size could be based on the size of the objects in the different regions.

For the system to be able to adapt to changes in the scene the classifiers should be constructed such that there is the possibility of updating them in an on-line fashion. This could be done by saving the extracted features in a buffer and after every fixed interval updating the model. To test this, a dataset with adaptive motion would have to be created, since no such exists.

Tests were done on the equal error rate, since this provided a method for comparing the methods. In a real-world scenario where there would probably be better to skew the error rates to get more false negative and less false positives. Tests could be done to see how many of the actual abnormalities the system picks up where one frame per abnormality would be enough. This should minimize the false positives and still output at least one frame of the abnormality.

Bibliography

[1] Advantages of surveillance cameras in schools. http://www.ehow.com/facts_5615866_advantages-surveillance-cameras-schools.html.

[2] Automatic number plate recognition. http://en.wikipedia.org/wiki/Automatic_number_plate_recognition.

[3] Axis. http://www.axis.com/.

[4] Chauvenet's criterion. http://en.wikipedia.org/wiki/Chauvenet's_criterion.

[5] Corner detection. http://en.wikipedia.org/wiki/Corner_detection.

[6] Crime prevention effects of closed circuit television: a systematic review. http://www.scribd.com/doc/8958655/Crime-Prevention-Effects-of-Closed-Circuit-Television.

[7] Emgu. http://www.emgu.com/wiki/index.php/Main_Page.

[8] Forbedret trafikinformation i aalborg med ny its. http://www.aalborgkommune.dk/Nyheder/Presse/2011/Sider/forbedret-trafikinformation.aspx.

[9] Geovision. http://www.geovision.com.tw/english/index.asp.

[10] Hog. http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients.

[11] How many cameras in the uk? http://www.securitynewsdesk.com/2011/03/01/how-many-cctv-cameras-in-the-uk/.

[12] Maryland department of transportation. http://www.traffic.md.gov/.

[13] Milestone system. http://www.milestonesystem.com.

[14] Optical flow. http://en.wikipedia.org/wiki/Optical_flow.

[15] Outlier. http://en.wikipedia.org/wiki/Outlier.

[16] Overvågning sker uden kontrol. http://www.dr.dk/Nyheder/Indland/2012/03/29/074624.htm.

[17] Pearsons product moment correlation coefficient. http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient.

[18] Sebastian thrun: Lecture 8 optical flow, feature tracking, normal flow. http://www.google.dk/url?sa=t&rct=j&q=lucas%20kanade%20eigenvalues&source=web&cd=4&ved=0CGoQFjAD&url=http%3A%2F%2Frobots.stanford.edu%2Fcs223b04%2FCS%2520223-B%2520L9%2520Optical%2520Flow.ppt&ei=YhSxT-vCFqXT4QTJ8OTYCQ&usg=AFQjCNEc0tJ9hlO-UNGSuKcEAQ6PwaMwzw.

[19] Sift. http://en.wikipedia.org/wiki/Scale-invariant_feature_transform.

[20] Silverman's rule of thumb. http://fedc.wiwi.hu-berlin.de/xplore/ebooks/html/spm/spmhtmlnode15.html/.

[21] Surveillance image. http://www.webcam-spy.com/text/video-surveillance.jpg.

[22] Take breaks or you make mistakes. http://therestdoctor.wordpress.com/2011/02/28/take-breaks-or-you-make-mistakes-22811/.

[23] Three sigma rule. http://en.wikipedia.org/wiki/68-95-99.7_rule.

[24] Tv-overvågning: Fakta om tv-overvågning i danmark. http://www.dkr.dk/sites/default/files/dkr_mat_083.pdf.

[25] Ucsd anomaly dataset. http://www.svcl.ucsd.edu/projects/anomaly/dataset.html.

[26] Video surveillance and monitoring. http://www.cs.cmu.edu/~vsam/.

[27] Video tracking. http://en.wikipedia.org/wiki/Video_tracking.

[28] Who' s watching? http://www.nyclu.org/pdfs/surveillance_cams_report_121306.pdf.

[29] Amit Adam, Ehud Rivlin, Ilan Shimshoni, and Daviv Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. IEEE Trans. Pattern Anal. Mach. Intell., 30:555–560, March 2008.

[30] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm, 2000.

[31] G. Bradski. The OpenCV Library. Dr. Dobb's Journal of Software Tools, 2000.

[32] M.D. Breitenstein, H. Grabner, and L. Van Gool. Hunting nessie - real-time abnormality detection from webcams. In Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, pages 1243 –1250, 27 2009-oct. 4 2009.

[33] Neil Bruce, D.B. and Pierre Kornprobst. Harris Corners in the Real World: A Principled Selection Criterion for Interest Points Based on Ecological Statistics. Research Report RR-6745, INRIA, 2008.

[34] Ke-Xue Dai, Guo-Hui Li, and Ya-Li Can. A probabilistic model for surveillance video mining. In Machine Learning and Cybernetics, 2006 International Conference on, pages 1144 –1148, aug. 2006.

[35] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. ARTIFICAL INTELLIGENCE, 17:185–203, 1981.

[36] I. Ivanov, F. Dufaux, T.M. Ha, and T. Ebrahimi. Towards generic detection of unusual events in video surveillance. In Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, 2009. AVSS '09., pages 61 –66, sept. 2009.

[37] P. Kaewtrakulpong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In Proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems, volume 5308, 2001.

[38] Jaechul Kim and Kristen Grauman. Observe locally, infer globally: A space-time mrf for detecting abnormal activities with incremental updates.

[39] Tai Sing Lee. Image representation using 2d gabor wavelets. IEEE Trans. Pattern Analysis and Machine Intelligence, 18:959–971, 1996.

[40] Liyuan Li, Weimin Huang, Irene Y. H. Gu, and Qi Tian. Foreground object detection from videos containing complex background. In Proceedings of the eleventh ACM international conference on Multimedia, MULTIMEDIA '03, pages 2–10, New York, NY, USA, 2003. ACM.

[41] Chen Change Loy, Tao Xiang, and Shaogang Gong. Stream-based active unusual event detection. In Proceedings of the 10th Asian conference on Computer vision - Volume Part I, ACCV'10, pages 161–175, Berlin, Heidelberg, 2011. Springer-Verlag.

[42] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[43] V. Mahadevan, Weixin Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 1975 –1981, june 2010.

[44] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. pages 935 –942, june 2009.

[45] V. "Reddy, C. Sanderson, and B.C." Lovell. "improved anomaly detection in crowded scenes via cell-based analysis of foreground speed, size and texture". In MLVMA11, pages "55–61", "2011".

[46] Vikas Reddy, Conrad Sanderson, Andres Sanin, and Brian C. Lovell. Adaptive patch-based background modelling for improved foreground object segmentation and tracking. In Proceedings of the 2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS '10, pages 172–179, Washington, DC, USA, 2010. IEEE Computer Society.

[47] Rene Schuster, Roland Mörzinger, Werner Haas, Helmut Grabner, and Luc Van Gool. Real-time detection of unusual regions in image streams. In Proceedings of the international conference on Multimedia, MM '10, pages 1307–1310, New York, NY, USA, 2010. ACM.

[48] Jianbo Shi and C. Tomasi. Good features to track. In Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on, pages 593 –600, jun 1994.

[49] Bin Zhao, Li Fei-Fei, and Eric P. Xing. Online detection of unusual events in videos via dynamic sparse coding. In The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, June 2011.

Part I

Appendix

Chapter 8

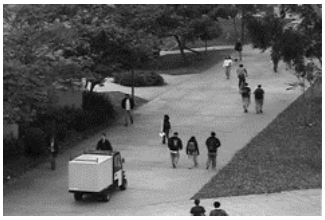
UCSD anomaly detection dataset

The dataset used for testing motion, size and texture abnormalities is the UCSD dataset [25] which is also used for testing in the paper which this project is based on.

It is described on the authors website as:

> "The UCSD Anomaly Detection Dataset was acquired with a stationary camera mounted at an elevation, overlooking pedestrian walkways. The crowd density in the walkways was variable, ranging from sparse to very crowded. In the normal setting, the video contains only pedestrians. Abnormal events are due to either:
>
> - the circulation of non pedestrian entities in the walkways.
> - anomalous pedestrian motion patterns.
>
> Commonly occurring anomalies include bikers, skaters, small carts, and people walking across a walkway or in the grass that surrounds it. A few instances of people in wheelchair were also recorded. All abnormalities are naturally occurring, i.e. they were not staged for the purposes of assembling the dataset. The data was split into 2 subsets, each corresponding to a different scene. The video footage recorded from each scene was split into various clips of around 200 frames.
>
> Peds1: clips of groups of people walking towards and away from the camera, and some amount of perspective distortion. Contains 34 training video samples and 36 testing video samples.
>
> Peds2: scenes with pedestrian movement parallel to the camera plane. Contains 16 training video samples and 12 testing video samples.
>
> For each clip, the ground truth annotation includes a binary flag per frame, indicating whether an anomaly is present at that frame. In addition, a subset of 10 clips for Peds1 and 9 clips for Peds2 are provided with manually generated pixel-level binary masks, which identify the regions containing anomalies. This is intended to enable the evaluation of performance with respect to ability of algorithms to localize anomalies."

(a) Training frame.

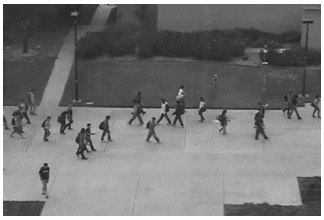


(b) Test frame with abnormality: golf car.

(c) Test frame with abnormality: wheelchair.

(d) Test frame with abnormality: bicycle.

Figure 8.1: Ped1: A sample training frame and three test frames.



(a) Training frame.

(b) Test frame with abnormality: bicycle.

(c) Test frame with abnormality: golf car.

(d) Test frame with abnormality: skater and bicycle.

Figure 8.2: Ped2: A sample training frame and three test frames.

Chapter 9

Direction datasets

9.0.1 Synthetic dataset

To test the concept of this method, synthetic video data has been created using MATLAB. Ten times five white squares of 15x15 pixels travels either up, down, left or right during 300 frames as in figure 9.1. This could also be made with highly textured squares which would improve the correct detection rate since the aperture problem is large when no texture is present, but for a concept test this should suffice.

If the region is a part of the foreground the summed optical flow will be saved to a text-file, where each region is represented by the four dimensional vector. This will be compared to the actual movement of the white squares.

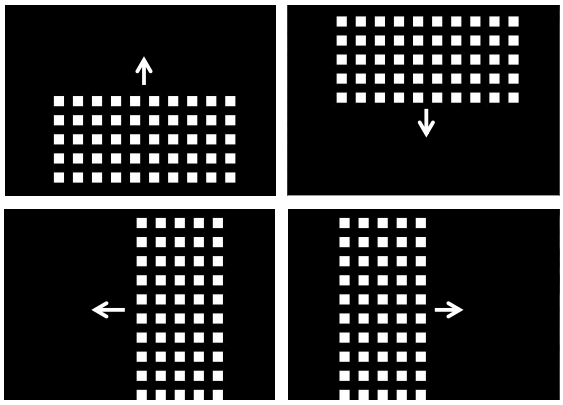


Figure 9.1: Figure of the four synthetic videos for direction concept test.

9.0.2 Real world dataset

To test the direction method on a real world scenarios a dataset containing direction abnormalities must be used. Since this does not seem to already exist a dataset have been created.

Many hours of traffic cam footage has been recorded from [12], which is of an American highway in Maryland. The refresh rate is captured in 5 fps which is enough for the optical flow to be able to track pixels. Furthermore the camera is fixed as described in the requirement specification 2.12.

Direction based anomalies, such as a car driving the wrong direction, happens very rarely. Therefore an hour of video has been edited and 15 small sections of the video has been reversed. This will ”simulate” a car or cars driving in the wrong direction.

A frame from the dataset can be seen in 9.3.

Figure 9.2: A frame from the direction dataset.

The abnormalities are distributed randomly over the entire range of the video. The length of each sequence spans from 25 frames (5 sec) to 125 frames (25 sec).

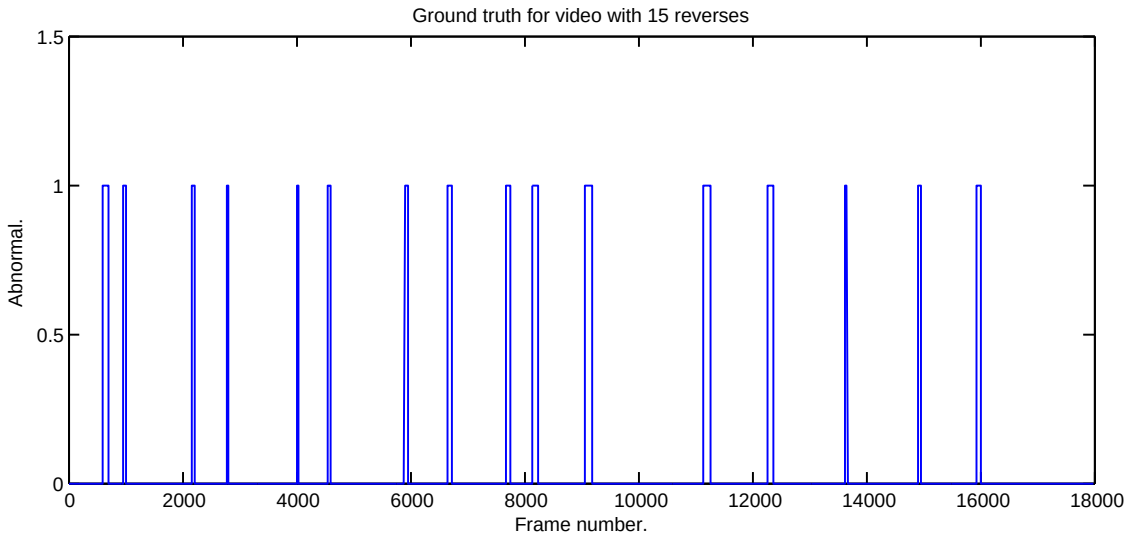


Figure 9.3: Ground truth for video. A value of one indicates an anomaly.

The classifier is trained by a 599 frames (2 min) sequence. This sequence contains no abnormalities.

Chapter 10

Survey

Title: Hunting Nessie – Real-Time Abnormality Detection from Webcams [32]
Author(s): Michael D. Breitenstein, Helmut Grabner and Luc Van Gool
Location: Computer Vision Laboratory - ETH Zurich
Year: 2009
Published: IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)
Type: Paper
Quote(s) according to Google Scholar: 14
Keywords: Incremental learning techniques, real-time abnormality detection, unsupervised method, data-driven, data structures, HoG, meaningful nearest neighbour, clustering.
Assumptions: Static camera, low frame-rate data.
Summary: They focus on situations where tracking and object detection is difficult due to the objects of interest being small, the frame-rate quite low (time-lapse like footage) and image quality also low. Therefore the data is treated as a stream of separate images focusing on the spatial information in each image. The basic idea is that everything not observed in the past is unusual. They discriminate between unusual scenes (never seen before) and rare scenes (similar scene seen before, but not very often).

A model is learned using the previous observed data. When a new image is captured a similarity measure is calculated that compares it with the learned model. Different similarity measures can be used, they use normalized cross correlation. Since the representation of the image has a high dimensionality most points are equally distant to a query point. Therefore concept of meaningful nearest neighbours is employed. To do this they store distribution of the similarities. If it is unlikely that the similarity score of a new image is drawn from this distribution, then the image is rated as unusual.

Observed data spans a high-dimensional space but is only sparsely populated. Therefore a transformation of data to a lower dimensional space is proposed. After transformation clustering techniques is used to choose representative data as cluster centers. This requires much less memory than storing all the data. To test if the chosen cluster centers represent the data well, they employ a *mean of the maximal similarity* (MMS) criterion. A threshold is used to make sure the MMS is high enough and therefore represent the data well. To make cluster centers diverse enough they minimize the absolute value of the threshold parameter withdrawn from the MMS. This results in a transformation that turns the image data into a hierarchy of extracted features. In the minimization problem the number of cluster centers is usually fixed because it is restricted by the memory. They choose quad-tree struce as the feature hierachy.

In the specific implementation they use Histogram of oriented gradients (HoG) as image features. They use a agglomerative clustering algorithm. When the number of observed data surpasses the

number of cluster centers they start deleting the weaker centers. They base this deletion of the age and number of times the center was a best match.

The system is divided into two parts. A maintenance phase and a detection phase. In the maintenance phase the model is updated and the detection phase the incoming data are classified.

They initially test on synthetic data. Here they show that basic functionality of the system performs as it should. Then they perform qualatively testing on footage recorded from public webcams. There is footage from two different locations. Time square and Loch Ness and it span several months. There isn't ground truth asociated as well as the number of false alarms and false positives are not stated. Rather some of the results where the system detects unusual events are highlighted and then an explanation for unusualness is sought in the data. How promising the results may look there is no hard statistics.
Comments: No temporal information is represented with this method. There might be a problem with the testing being too much of anecdotal nature. The results are not thoroughly documented with ROC or confusion matrix.

Title: Real-time detection of unusual regions in image streams [47]
Author(s): R. Schuster, R. Mörzinger, W. Haas, H. Grabner and Luc Van Gool
Location: Joanneum Research, Austria. Computer Vision Laboratory, ETH Zurich, Switzerland.
Year: 2010
Published: Proceedings of the international conference on Multimedia
Type: Paper
Quote(s) according to Google Scholar: 1
Keywords: INcremental learning techniques, real-time abnormality detection, unsupervised method, HOG, meaningful nearest neighbour, clustering. **Assumptions:** Static camera, low frame-rate data. **Summary:** As with Hunting Nessie they focus on low frame-rate data where optical flow or other methods cannot be used. They utilise the HOG descriptor to describe each region and a database classification system where each entry is a HOG descriptor. When a descriptor is introduced into the classifier the distance from the input descriptor to the nearest entry in the database is measured. If the nearest distance is greater than the inner model distance of the database classifier the two nearest entries are merged and a new is created with the newest input. There is an upper limit to how many entries there can be in the database, to keep the computational requirements to low. This is regulated by a constant which can be set differently. The method achieve 20 fps. **Comments:** The method works quite well for detecting out of place objects and runs in real-time. There is an unsupervised learning algorithm which is optimal and it is data-driven so no manually thresholds have to be set.

Title: A Probabilistic Model for Surveillance Video Mining [34]
Author(s): Ke-Xue Dai, Guo-Hui Li and Ya-Li Can
Location: National University of Defense Technology, Changsha 410073, China
Year: 2006
Published: Proceedings of the Fifth International Conference on Machine Learning and Cybernetics
Type: Paper
Quote(s) according to Google Scholar: 1
Keywords: Background subtraction, expectation-maximization, hidden Markov model
Assumptions: Not clear, number of clusters known in advance
Summary: They perform basic background subtraction to detect pixels where movement has occured. For each frame a binary image is determined by subtracting the background image from the current and then threshold the results. They then calculate a motion scalar for each by summing all the pixels of the binary image that represents pixels where motion has occured. They have some different rules for updating the background image mainly based on different learning rates.

They use expectation maximization to learn the parameters of a Hidden Markov Model (HMM). Tests are performed on footage they recorded. No information on the length of the footage or the kind of events in it is described in the paper. They note that a classification accuracy of 75% is achieved.
Comments: The paper is so badly written that it is very difficult to understand.

Title: Online Detection of Unusual Events in Videos via Dynamic Sparse Coding [49]
Author(s): Bin Zhao, Li Fei-Fei and Eric P. Xing
Location: Stanford Vision Lab
Year: 2011
Published: IEEE Computer Vision and Pattern Recognition (CVPR)
Type: Paper
Quote(s) according to Google Scholar: 0
Keywords: Sparse coding, unsupervised, online
Assumptions: Video data
Summary: The basic idea of this paper is to learn and initial dictionary of events. Each event is then reconstructed using the dictionary. Based on how well the reconstruction can be performed an object function is calculated. The lower the number of this function the more likely it is that en event is normal. Also the dictionary is updated after each event.

First the event description is extracted. A sliding window is employed along the spatial and temporal axes. In each window spatio-temporal interest points are detected. These interest points are described with HoG and HoF feature descriptors. A group of spatio-temporal cuboids residing in a sliding window define an event. They then formulate a sparse coding problem. An object function is defined which takes an event, reconstruction weight vectors α and the dictionary $\mathbf{D}$ as input. The lower the value of the object function the more likely and event is normal. The dictionary represents what is normal and consists of a number of base vectors. The intuition is that an normal event can be reconstructed using a small number of these bases. The object function has three terms. One that measures how good the event can be reconstructed from the dictionary (Reconstruction error). The second measures how many of the bases needs to be used (Sparsity regularization). The third measures the similarity of nearby cuboids (Smoothness regularization). The minimization problem of finding $\mathbf{D}$ and α is not jointly convex, but is convex for each of them. They alternate between minimizing the object function for $\mathbf{D}$ and α which then converge to a local minima. For each event the reconstruction weight vectors that minimizes the object function are found. If the object function is larger than a given threshold the event is categorized as unusual. The dictionary is updated using stochastic gradient descent after each event. Test are performed on surveillance videos of subway exit (43 minutes) and a subway exit (1 hour 36 minutes), in both cases a static camera. This data is not a standard public dataset, but has been used by others and results are compared. Furthermore they test on 8 youtube videos which have different camera motions. [1] In the tests they use a sliding window if size 80x80 along the x and y axes and 40 frames along the time axis. In the subway exit footage they use the first 5 minutes to find the initial dictionary. This takes 20 minutes and unusual event detection takes 2 seconds pr frame.
Comments: There are very few assumptions and parameters that need to be set. They formulate the problem as minimization problems which means that some more standard method can be used. They might not be able to detect longer term unusual events since the cuboids used only span 40 frames which is less that two seconds assuming a frame rate of 25 Hz. Our untuition says that an event needs to show unusualness on this scale to be detected by this approach. This longer lasting unusual event does not seem to be considered in the paper.

Title: Towards Generic Detection of Unusual Events in Video Surveillance [36]
Author(s): Ivanov, I., Dufaux, F., Ha, T.M. and Ebrahimi, T.

[1] Some of the youtube videos can be seen here: `http://sites.google.com/site/binzhao02/home/cvpr_2011_video`

Location: Ecole Polytechnique Fédérale de Lausanne, Switzerland
Year: 2009
Published: Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on
Type: Paper
Quote(s) according to Google Scholar: 6
Keywords: Feature extraction, moving object trajectory, support vector machine, unusual event detection, video surveillance, image motion analysis, supervised
Assumptions: Static camera, video data (25Hz), supervised learning, no camera calibration
Summary: The basic idea of this paper is to detect unusual events based on the acceleration and velocity of the objects in a scene. This is achieved by extracting trajectories of objects from video sequences. The trajectories are pre-processed in order to account for occlusions and noise. Bresenhams line-drawing algorithm is used to fill trajectories were occlusion has occured. A Savitzky-Golay filter is used to smooth the trajectories. The Savitzky-Golay filter tends to preserve relative maxima, minima and width. The velocities and accelerations are calculated from the smoothed curve. Each trajectory is sampled at M=128 equidistant points and from these the velocity and the accelation is extracted. This results in 256-dimensional feature vector. They then train a SVM-classifier and use that to classify test data as usual or unusual. The main strength of the approach is the ability to train the classifier on one scene and then use this on a completely different scene. This is due to the nature of the features that are based on accelerations and velocities in the trajectory. Therefore it will detect activity that exhibit abnormal velocities or accelerations. Finally the system is tested with good results on the PETS dataset. The authors own assesment of the contribution: *The main contribution of this paper is the flexibility of the used feature representation which adapts to different situations. Namely, it is possible to train the proposed system with normal and unusual trajectories from one sequence and to test it with different sequences from completely different scenes. Detection of unusual events in terms of velocity and acceleration is possible due to velocity invariance achieved with different kinds of scaling (normalization).*
Comments: Gives a nice overview over the building blocks of a trajectory based unusual event detection. The basic trajection extraction is not the focus of this paper and is described in another paper by one of the authors.

Title: Anomaly Detection in Crowded Scenes [43]
Author(s): Vijay Mahadevan, Weixin Li, Viral Bhalodia and Nuno Vasconcelos.
Location: San Francisco, CA
Year: 2010
Published:
Type:
Quote(s) according to Google Scholar: 48
Keywords:
Assumptions: High frame-rate, regions
Summary: This method combines both spatial and temporal anomaly detection. There is a training and a testing phase, so they do not use on-line updating of the normal model.

For the temporal anomalies an MDT [43] is computed on each frame in the training set and in the running phase the computed MDTs are compared by using log likelihood for the cluster that is closest to the MDT. This will produce the temporal anomaly map.

For the spatial anomalies saliency method is used. This is a measure of how much the region stands out of its surroundings. An MDT is computed using an approximation, since it would otherwise be too computationally heavy. The spatial anomaly map is computed by using KL divergence between the MDT of the training-set and the input frame.

The two anomaly maps are joined by adding them together and the anomaly detection is done using a simple threshold.

In the testing phase it takes 25s for each frame to be processed, which makes this method far

from real-time.
Comments:

Title: Improved anomaly detection in crowded scenes via cell-based analysis of foreground speed, size and texture. [45]
Author(s): Reddy, V.; Sanderson, C.; Lovell, B.C.;
Location: Colorado Springs, CO
Year: 2011
Published: 2011
Type:
Quote(s) according to Google Scholar: 4
Keywords: Region-based, optical flow,
Assumptions: Grayscale images, high frame rate
Summary: The article is an improvement of the MDT method proposed. The scene is split into regions of 16x16 where the anomaly detection is based on the output from each region. Foreground segmentation is done using their own algorithm which provides a stable foreground mask in crowded scenes.

Three features are used for classification and only the foreground pixels are used for computing the descriptors: Average optical flow where both spatial and temporal is used for averaging. Size is found by foreground occupancy in the region using the neighbouring regions with a gaussian mask. Texture is described by a 2D Gabor wavelet in four different directions.

For classification a cascade model is used. First the probability of the flow in each region is evaluated and if considered to be abnormal the region is classified as abnormal. If the flow is normal the size and texture will then be evaluated.

Finally noise is combated by evaluating evaluations of regions temporally. Only if the previous or later region or neighbouring region is evaluated to be anomalous then will the region be classified as anomalous.

On the UCSD dataset[25] it outperforms Social Force, MPPCAand MDT while running at 12fps on a standard 3Ghz computer.
Comments:

Title: Robust Real-Time Unusual Event Detection Using Multiple Fixed-Location Monitors [29]
Author(s): Adam, A.; Rivlin, E.; Shimshoni, I.; Reinitz, D.;
Location: Technion-Israel Inst. of Technol., Haifa
Year: 2008
Published: 2008
Type:
Quote(s) according to Google Scholar: 82
Keywords: Optical flow, regions,
Assumptions: High frame-rate.
Summary: Using multiple fixed-location monitors is about extracting information from regions and evaluating their normality. The optical flow for each location is computed using the Lucas-Kanade method and both direction and velocity can be taken into consideration, depending on the scene. For each monitor the pdf (probability density function) is found and represented as a histogram if they are evaluated to be representative of the monitor.

The histograms are saved in a fixed-length buffer which automatically discards old observations. For each new direction or velocity monitor, the likelihood is found by comparing the it to the entries in the buffer. Furthermore, to overcome false positives, temporal integration is used. If a fixed amount of frames produce alarms consecutively the system will output an alarm. This proves to be very effective.

The method produces very good results for detecting people going the wrong way in a subway and for people running in a supermarket. Unsupervised learning is used, but a few parameters

needs to be fixed during the setup. Normally either direction or velocity flow is used because of the high computational cost of both.

Comments: The method is outperformed by [45] which is also running with acceptable fps.

Title: Stream-based active unusual event detection [41]
Author(s): Chen Change Loy, Tao Xiang and Shaogang Gong
Location: School of EECS, Queen Mary University of London, United Kingdom
Year: 2010
Published:
Type:
Quote(s) according to Google Scholar: 4
Keywords: Optical flow, regions, Bayesian classifier.
Assumptions: High frame-rate.
Summary: Optical flow is computed for each pixel in each region using the Lucas-Kanade methods [42]. The motion in each region is quantised into four directions and the motion in the next frames are summed up to make a four-bin histogram containing the motion of 50 frames. This will then be converted to a codebook of 16 possible combinations ranging from no significant motion in either of the 4 directions to significant motion in all of the four directions.

The classifier is a naive Bayes classifier and a threshold is used for determining if a frame is abnormal or not.
Comments:

Chapter 11

Parameters used in program

Parameter	Note	Static/Dynamic	Default	Range
Resize factor	The resize factor for input images or video	static	1.0	-
Region height	The height of each region	dynamic	16	8;16;32
Region width	The width of each region	dynamic	16	8;16;32
Min samples for pmf	The minimum pixels for generating pmf	dynamic	as histogram bins	5;10;20;30
Histogram bins	The number of bins in histogram	dynamic	20	5;10;20;30
Texture threshold	The threshold for texture classifier	static	0.9	-
Motion threshold	The threshold for motion classifier	dynamic	-	0-0.3
Size threshold	The threshold for size classifier	dynamic	-	0-0.3
Direction threshold	The threshold for direction classifier	dynamic	-	0-2.0
PyrLK: number of features	The maximum number of features for "good features to track" to find	static	1000	-
PyrLK: number of pyramids	The number of Gaussian pyramids used	static	2	-
PyrLK: stop criterion	The number of max iterations	static	100	-
PyrLK: window size	Search window size	static	15 x 15	-
LK: window size	Search window size	static	15 x 15	-
HS: stop criterion	The number of max iterations	static	20	-
FGS: min area	Minimum area for foreground segment algorithm	static	0	-
FGS: perform morphing	Number of erode/dilate operations	static	0	-