Behaviour classification in urban area using video based sensor networks

A Thesis Submitted for obtaining
the Scientific Title of PhD in Engineering
from
Politehnica University Timișoara
in the Field of Computers and Information Technology
by

Eng. Marius BABA

PhD Committee Chair:
PhD Supervisors: prof. dr. ing. Ionel JIAN
Scientific Reviewers:

Date of the PhD Thesis Defense:

LIST OF FIGURES

List of tables

LIST OF TABLES

CONTENTS

Contents

1 Introduction

This chapter aims to describe the topic of this PhD thesis, namely the Tracking Solution in Video-Based Sensor Networks. The scope of this research work is to propose a video surveillance system that is able to detect important events in the urban areas. The sisyem relies on using computer vision algorithms that do not demand high computational power. Therefore , my primary focus was to design computer vision algorithms that use as litle computation resources as possible. In other words I aimed to design not computationaly intensive algorithms that operate in real time and are capable of analysing complex scenes so the are able to detect human behavio in video. This strategy allows the surveillance camera to process the video stream on its own, so the process of detecting behavior in a large surveillance system is much simplified.

As we all know, surveillance systems in big cities are composed of many surveillance cameras that are connected to several central stations. The cameras usually do not include any processing, but instead transmit video to the central stations via the network for further analysis. This architecture requires a powerful computing power of the central stations and also a fast and reliable network to be able to transmit data in real time. In figure 1 is an example of such architecture.

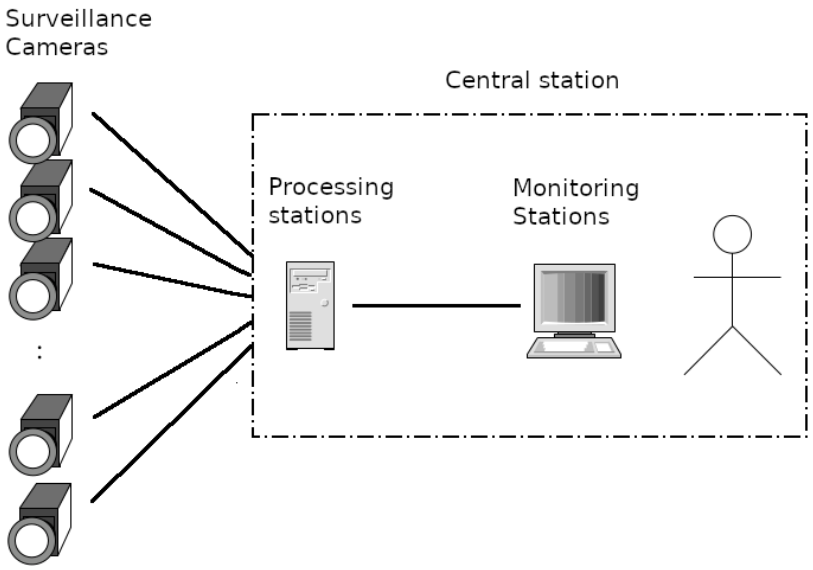


Figure 1 - Centralized video surveillance architecture

Therefore, incorporating video analysis processing into surveillance cameras reduces the need for a powerful central station in terms of computational power and also reduces the volume of data traffic on the network. Instead of streaming the video data to the central station, the surveillance cameras send only small data packets containing the results of the video analysis process. Such an distributed processing architecture is much more convenient especially for the batery powered surveillance systems which are designed to be deployed in zones where there is no video surveillance infrastructure.

The video data is analysed individually by each surveilance camera node and the results are transmitted via the wireless network to the central station. In this way, the data no longer has to be processed by the central unit and thus the central unit does not have to have high computing power. Such a surveillance system is illustrated in figure 2.

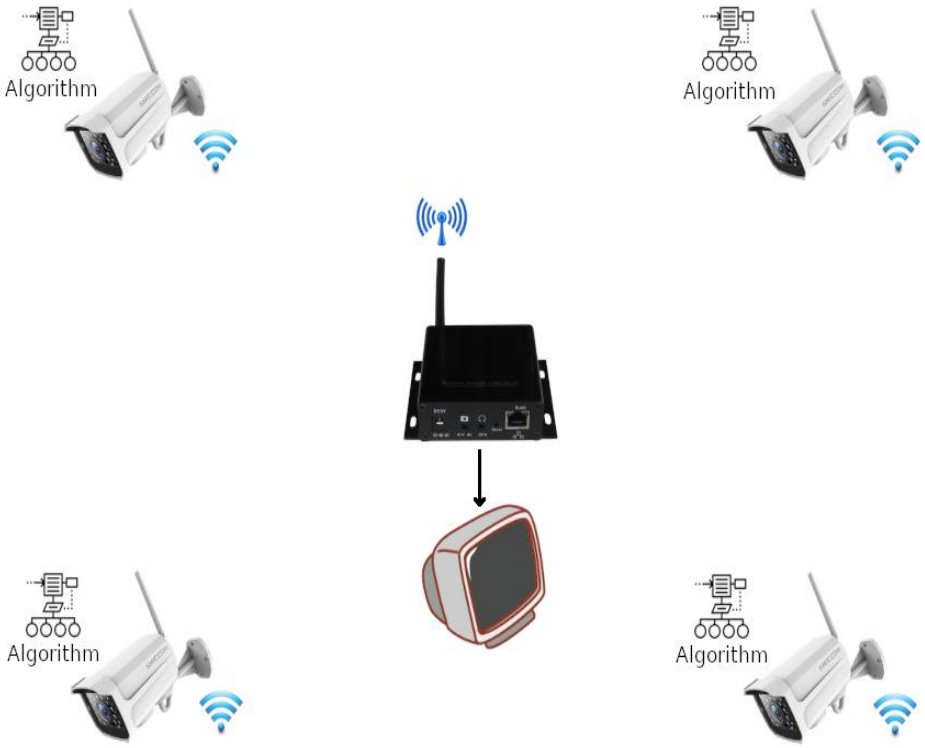


Figure 2 - Wireless surveillance network with embeded video analysis into the surveillance camera

This architecture also has another advantage. Because the cameras do not require a cable connection, they can be, for example, battery-powered devices. This type of surveillance system is suitable for use in cases where a quick installation of the surveillance system is required. For example, to monitor protests or music concerts.

However, the most sensitive part of this system is the video analysis algorithm. It is application oriented and requires a lot of design attention, as it is expected to achieve good detection performance and to run on low resource hardware. Moreover, designing a video analysis algorithm that is meant to detect behavior is not an easy task. Behavior understanding is a complex process that oftentimes is misunderstood even by the humans. Therefore, this process must be analyzed in great detail in order to design a successful algorithm.

2 Why human behavior understanding is important

In this section I am going to tackle the reasons why understanding the human behavior is so important.My personal opinion is that human behavior is a very interesting field to study. Behavior of animals was exploited by the humans from the beginning of our era. In the need for food the humans hunted animals for survival. The hunting process involved tracing the victims and catching them in a trap. For efficient hunting the hunter needs to know the victim next move based on the presented clues. In other words, the hunter needs to be familiar with victim’s behavior.

Same scenario applies now-days except that the behavior analysis is extended to many other areas like biology (analyzing the behavior of a cell), sociology (analyzing human behavior), medicine, civil engineering, traffic management and many others. Although the idea is old it is exhaustively exploited in many modern science fields for ever since.

As an example in medicine, in order to establish the patient's diagnosis, doctors have to perform a lot of tests, thus obtaining a lot of data regarding patient behavior. By combining these data with the doctor's knowledge, a diagnosis can be established. Such a diagnostic technique is used to detect many diseases like Parkinson and Alzheimer. Similarly, this technique is also used for the early detection of children with intellectual disabilities. Early detection of the disease is very helpful for patients because doctors can more easily control it.

Similarly, in sports, it is common practice for coaches and managers to analyze the recorded videos of opponent teams. With the help of this analysis, they are able to

understand the behavior of their opponents and thus can anticipate certain unfavorable actions for their team that have very high chances to occur during the football match. This helps the coaches to better plan their game and thus their team has higher chances to win the game.

Another example where behavior analysis plays an important role is the analysis of the employee behavior. In this sense, the psychological study elaborated in [1] uses the analysis of employee behavior to demonstrate that there is a direct link between the age of employees and their attitude towards work. Attitude towards colleagues and other values such as job satisfaction, preferences and motivation are all analyzed through employee behavior. Through this technique, the authors have shown that there is indeed a link between the age of the employee and his interest in work. This information can be very helpful for company managers. They can assign tasks based on this finding, so that the work is done more efficiently.

Similarly, the article elaborated in [2] investigates the behavioral patterns that can tell us whether an employee has a carrier or group oriented commitment. More precisely, the authors collected the data from 287 employees by using self reports. So, by using the acquired data they managed to develop a method that can successful classify the employee's orientation. This type of information can help manager better plan tasks. So each employee receives tasks that suit him best.

The technique of recognizing human behavior can also be used to estimate the next action of individuals in urban areas. This estimation is particularly important in scenes that involve dangerous human actions like fight, robberes and other similar activities. If actions are correctly estimated, such unwanted human behaviors can be stopped. That is, this approach allows for the early detection of dangerous behaviors that can be used, for example, by the police in order to get to the scene very quickly.

2.1 Why automation of human behavior understanding is so challenging

Analyzing people's behavior is a complex task for computers. In order to recognize human behavior, the computers need to analyze a lot of data and to recognize many patterns in it. People are able to more easily recognize behavior of other people, but computers do not. Generally speaking, the human behavior can be described as a collection of human actions. Of course, there are exceptions to this rule, in some cases the behavior may consist of just one long action.

According to the literature, there are several levels of abstractions for the human behavior that are used in computer vision algorithms for detecting human behavior. The taxonomy proposed in [3] classifies the behavior analysis tasks based on the

degree of semantics and the amount of time required for the task to complete as illustrated in the figure3.

Figure 3 - Classification of the behavior analysis tasks [3]

Therefore, at very basic level is the motion detection which includes background subtraction and motion segmentation. The next level includes action detection in winch the human motion is recognized. This task aims to detect the type of action, like running, walking, siting, as well as the person's interaction with the objects in the scene. After actions, follow the activities in which human behavior is detected based on the analysis of several human actions. This level consists of analyzing the scene for a very short period of time. The analysis usually takes a few seconds to a few minutes. Therefore, this category includes behaviors such as showering, cooking, eating and so on. The level with the highest semantics is labeled as behavior. It is located at the top of the pyramid in the figure3. The behavior is defined by authors as a collection of actions that are captured over a longer period of time that is hours, days, or even weeks. In this category belong behaviors like personal habits, daily routines and other similar activities.

Similarly, the authors in [4] define an action taxonomy based on three levels of abstraction. In this sense, they introduce the primitives of action, the action, and the activity. Action primitives include basic human movements that are performed at the limb level. A collection of such action primitives forms an action while the activity consists of a collection of actions involving interactions between humans and objects. For example, the process of making tea includes arm movements that are action primitives, taking the glass from the table is the action itself, and since the whole process involves multiple actions and interaction with objects, brewing tea is defined as an activity.

Nevertheless, action recognition plays a key role in recognizing human behavior. Many times, it is hard to predict the behavior of an individual or of a group. The

main problem is that people can have quite different actions for a certain behavior. That is, the same behavior does not necessarily mean the same set of actions.

2.2 Behavior understanding in video surveillance

Intelligent video surveillance systems like in thefigure 4.a are becoming increasingly important along with today's problems, such as terrorist attacks, natural disasters caused by global warming and cities overcrowding. Almost all solutions in this field are based on video analysis algorithms which aim to capture important activities like human actions, human behavior or traffic surveillance events. The purpose of such systems is to help or replace the painstaking work of surveillance officers figure 4.b, who face the problem of monitoring multiple screens at once for long period of time. Their job is to analyze each video surveillance stream and identify irregularities in it. This is where the problem arises.

According to medical experiments, human eyes get tired very easily, especially if they constantly change their focus of attention. The authors in [5] have demonstrated that human can easily omit actions in a video. They performed an experiment in which a man had to detect all human actions in a thirty-minute video that was played on a single screen. Upon completion, they found that he had failed to detect more than 80% of the actions. This is a very clear indicator that humans are not efficient in analyzing surveillance videos. To remedy this problem, an automatic video analysis system would be much more suitable for such a task.

a) b)

Figure 4 - Surveillance system in urban areas; a) surveillance cameras; b) camera control center

Nevertheless, another problem that automatic surveillance systems help a lot is due to the large amounts of data provided by video surveillance systems in the city. As surveillance cameras become cheaper, more and more cities are equipped with surveillance systems consisting of hundreds or thousands of cameras. This fact creates a new problem. The increased number of cameras in the cities are able to capture a lot of information that is unable to process by the surveillance officers. Therefore, only part of it is analyzed in real time, and the rest is stored in memory

for on demand analysis. However, this is not a very efficient solution because many important events are missed. A better option is to use an automated video analysis system so that video surveillance streams are quickly analyzed by computers.

Such computer vision systems are popular in various fields. For example, they are frequently used in medicine for the task of healthcare monitoring of elderly people [6]. In these scenarios the algorithms are used to recognize human actions, human behavior, and human incidents. This information is very useful because it can reveal mental and physical problems of the monitored patients. In figure 5is an illustration of a fall incident that is common in a nursing home. These types of incidents are usually detected late by surveillance officers. So, the patient can stay in this state for several tens of minutes or even hours. For this reason, his health may worsen. To avoid this situation, it is much more efficient to use a video analysis algorithm that constantly monitors video surveillance streams and alerts the surveillance officers if something goes wrong.

Figure 5 - Four fall incidents captured by surveillance cameras in a nursing home [7]

Another case that surveillance officers cannot cope with is the detection of unusual events in urban areas. Due to the large number of surveillance cameras, the officers miss many important events such as loitering detection, intrusion detection and unsupervised baggage detection which adversely affects public safety. For this reason, urban surveillance systems use computer vision algorithms to increase the detection rate of such dangerous events.

Why human behavior understanding is important

These types of computer vision algorithms are designed to improve public safety by monitoring human behavior. For example, loitering is often associated with vulnerable people seeking attention. As such, they are considered a real threat to public safety because they can harm other people by physically attacking them or they can cause material damage by destroying things around them. The act of loitering means to hang out in a certain area for longer period of time, with no real purpose. Thefigure 6.a illustrates such an act of a person inside a waiting room which is successfully captured by the loitering detection algorithm [8]. Nevertheless, the immediate detection of loitering behavior is important to public safety because can prevent the occurrence of aggressive behavior.

Similarly, computer vision systems are frequently used in airports or railway stations by the security operators to early detect the unattended baggage act. Because there have been numerous incidents in recent years, caused by terrorist organizations that aimed to plant explosive devices with the help of luggage, this human behavior is desirable to be captured as soon as possible. Thus, the early detection of such unattended baggage act can prevent the destruction of transport infrastructure and they also save the lives of many people. The figure 6.bshows an unattended baggage from a train station that is captured by the computer vision algorithm proposed in [9].

a) b)

Figure 6 - Surveillance systems using computer vision; a) Loitering behavior of a person inside a waiting room[8]; b) Unattended baggage from a train station [9]

Computer vision algorithms can also aid surveillance officers in the detection of anomalies in crowded places. This scenario is often encountered in today's cities, as they become larger and larger. According to information gathered form [10] 55% of the world's population currently lives in urban areas. This number is constantly increasing, so that by 2050 it is estimated that the percentage will reach 68%, which means that the cities will get overcrowded. Increasing the number of people in a city also means increasing the crime rate. Therefore, the prevention of such events is essential for public security.

To deal with this problem, surveillance systems must constantly analyze many video streams. It is known that the probability of unusual behaviors increases with the number of people. Therefore, it is most likely that an unusual event will take place in crowded places. For this reason, the surveillance analysis should be focused on the crowded places of the city. In this sense the computer vision algorithms such as [11][12] may help the surveillance officers to detect unusual behavior in crowded areas.

These types of computer vision algorithms differ from the algorithms that detect actions of individuals. They analyse the group behavior and not the behavior of individuals. The reason for such an approach is that the actions of individuals causing problems are difficult to recognize in a crowd. Their bodies are usually occluded by other people's bodies and thus recognition of the action is difficult in such cases.

Examples of such unusual human behaviors that are common in a crowd are scenes in which people run in one direction or scenes in which people disperse from a central point due to panic, fight or clash. These scenarios are illustrated in figure 7. The first two pictures of figure 7.a fromthe left illustrate the running in one direction scenario while the last one shows an example of crowd dispersion. The examples are from the UMN dataset that is described in the article[13]. Another type of anomalous behavior that is common in crowds is the occurrence of unusual events in scene, such as the presence of cyclists, skateboarders or a vehicle in the pedestrian area. These scenarios are illustrated in the in figure7.b. They are from the UCSD dataset which is described in the article [14].

a)

b)

Figure 7 - Anomalous crowd behavior; a) samples from the UMN dataset [13]; b) samples from the UCSD dataset [14]

3 Thesis structure

In the following sections I am going to describe the achievements that I gained during my research period. Beside the revealed novelties, I am also going to describe the state of the art approaches that are related to my research. The last part of this manuscript points out my contributions. Nevertheless, the thesis is structured as follows.

The next chapter that is the chapter three is constrained about describing the state of the art solutions that are used in the computer vision applications for detecting human activities. More specifically these state of the art approaches are computer vision algorithms that are designed to automatically detect human events in video. Some of them are concerned about detecting simple human actions whilist others are aimed for detecting complex behaviors.My goal was to design a smart surveillance system that is able to improve the public safety so I investigated just computer vision algorithms that are designed to recognize abnormal human actions and unushual human behaviors. Theoretically these algorithms could be a perfect fit for a smart surveillance network that I propose in the chapter four,but all have their drawbacks so they can not be used in this sort of applications.

In the chaptapter four I propose a novel smart surveillance system.The first part describes the details of its architecture and also provides a brief description of the procedures for installing surveillance cameras. Beside system description the chapter four also contains the details of a computer vision algorithm I designed for the novel surveillance system.This novel algorithm was a first step towards designing a more advanced computer vision algorithm that could analyze complex urban scenes. In this sense, I designed my first video analysis algorithm that is capable to recognize simple behaviors that are encountered in the traffic surveillance applications.

The chapter five introduces a more advanced computer vision algorithm than the one described in chapter four. It uses an advanced technique that is based on motion understanding in order to classify human behavior in video.The algorithm is able to detect unushual human behavior in video streams that are captured by surveillance cameras. These types of events are important to detect because they can prevent dangerous situations from happening.

In a similar manner, the chapter six describes another novel computer vision algorithm that is an extension of the algorithm presented in chapter five. It serves for the same purpose that is to detect unushual human behavior in video, but compared to its brother it is able to analyze more complex scenes. The algorithm is also based on motion understanding, but in the detection process beside the classical computer vision techiques it also uses a deep learning techique. This hybrid

approach enables it to analyze complex scenes thus it can easely spot the abnormal behaviors of people in video.
It is important to note that all the computer vision algorithms that I propose in this thesis are meant for the novel smart surveillance system described in chapter four.

Lastly, the chapter seven contains an overview of my work. The conclusion section within this chapter is a brief resume of the approaches that I propose in this thesis. It is followed by a contribution sections in which I point out my contribution to knowledge.

4 State of the art in behavior recognition

In this chapter I am going to describe the available computer vision algorithms that are designed to capture human actions in video. Action detection is a valuable approach, because it enables us to recognize the human behavior in video. The first generations of action detection algorithms were manually crafted. This means that the feature extraction and the action classification consist of manually tuned algorithms, relying on human analysis and understanding of the problem at hand. Usually, these solutions only work in the environments for which they were designed and under the underlying assumptions made by the designer. These assumptions are made based on human analysis of the available application data. Due to design time constraints and possible memory flows of the designer, the effectiveness of this approach in complex problems is limited.

The second generation of action detection algorithms described in this chapter are based on the deep learning approach. Opposite to the handcrafted approach, the deep learning solutions are automatically derived from data. The amount of data that can be used for learning is not limited by the availability of time and the perseverance of the designer. Once defined, a deep learning architecture can be retrained and used for detecting the actions in many environments with just a little fine tuning. One downside of the deep learning solution is the need of considerably larger learning datasets than those used in handcrafted solutions.

4.1 Handcrafted video analysis solutions for action recognition

The handcrafted solutions presented in this chapter include state of the art video analysis algorithms that aim to either recognize human actions in video or recognize violent events. Violent events are nothing but certain types of actions, so both types of algorithms aim to achieve the same general goal.

Based on the features used, the handcrafted video analysis solutions can be divided into two categories:

- solutions that use holistic representations and
- solutions that use local features

Solutions that use holistic representations involve detecting and locating the human body. They usually use the human shape or silhouette as features to detect the action. Instead, the second category includes solutions that use only local features. The advantage of this method is that it does not require the explicit detection of the human body. It uses one or several feature extractor algorithms, in order to capture the human body information relevant for the analysis.

4.1.1 Solutions that use holistic representations

In search of a computer vision algorithm, that can run in real time on a smart surveillance camera and is designed to detect human behavior in video, I explored several state-of-the-art computer vision algorithms that use holistic representations.

One such promising algorithm that caught my attention is the one presented in [15]. It is able to detect human abnormal events from the urban surveillance videos. The algorithm uses several computer vision techiquies to achieve this task. Therefore, in the figure 8 is an illustration that shows its block diagram.

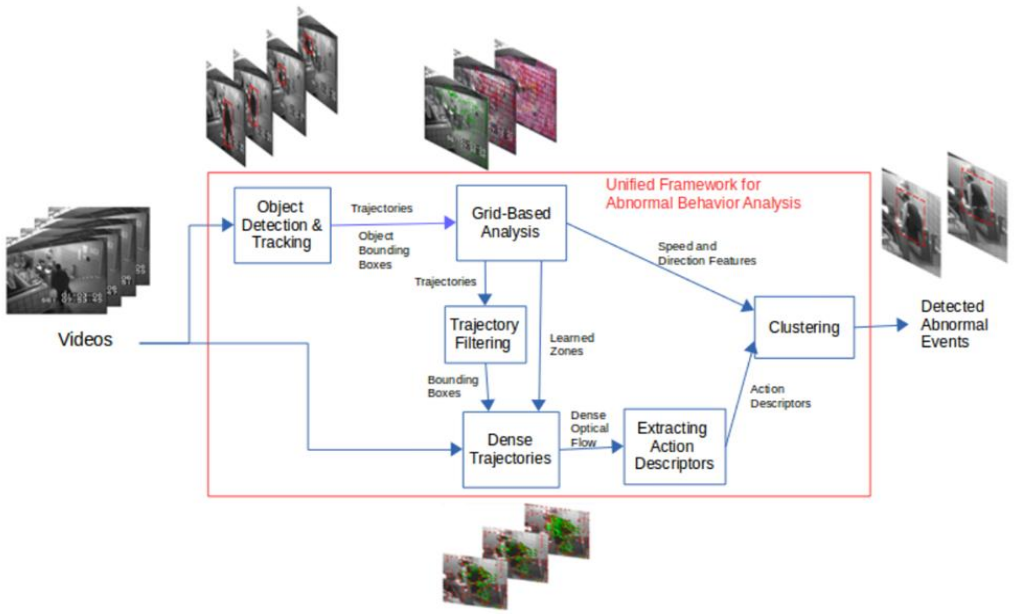


Figure 8 - The block diagram of the video analysis algorithm that aims to detect abnormal events of people in urban areas [15]

In the first phase the video scene is segmented by the object detection and tracking block of the algorithm. All humans in the video are recognized and their positionsare stored over time. In order to be able to priceisely estimate the object positions the approach uses a tracking algorithm that is propose in the article [16]. This method of detection seemed to be very effective, so I tried it in my experiments and decided to use a similar recipe in the algorithm I propose in chapter five.

Moreover, the proposed approach is able to detect groups by using trajectories in the scene. If there are two or more people next to each other, the algorithm groups them together and tracks the whole group as one object, by using the algorithm proposed in [17].

In order to reduce the processing time, the authors propose the elimination of the trajectory points that are not sufficiently descriptive for action detection. In the article, this process is called trajectory snapping. It consists of applying a grid over entire the frame and grouping all the trajectory points that are found within each grid element. Therefore, instead of having multiple points each grid element has just one most representative trajectory point. This process reduces the number of points in each trajectory.

The reduction of the trajectory points is followed by a discovery of the interest zones. Both phases are based on grid analysis. Clustering the zones in the videos makes it easier to recognize the violent activities. Thus, the algorithm detects areas where people stand still, areas where people move at a moderate speed and areas where people run. For clustering the zones, the authors decided to apply the affinity propagation algorithm [18] to the already extracted trajectories.

The next phase, namely the trajectory filtering, is responsible for detecting abnormal trajectories and separating them from normal ones. It implies the use of a Gaussian distribution to model the direction and speed of the trajectories. In the case when the trajectory direction or the trajectory speed is not found in the model it is considered as abnormal. This approach of trajectory filtering is able to detect only a small number of anomalous events in the scene. Despite of this limitation the approach is very effective in recognizing human actions. For this reason, I decided to adopt the trajectory descriptors used by this method in my algorithm that I propose in chapter five.

Hence, to further improve the violence detection performance, the authors include a spatio-temporal filtering phase. The process consists of extracting the trajectories of optical flow and the HOG, HOF, MBHx and MBHy descriptors for several consecutive frames. These descriptors extract the appearance and the motion of objects in the scene. Furthermore, because the extraction of the descriptors for entire frame introduces a processing overhead, the authors apply descriptor extraction only on

the bounding boxes provided by the tracking algorithm. So descriptors are computed only for zones of the scene that contain people.

For modeling actions using descriptors, the proposed approach uses the Bag of Words model. The actions are described by the histogram of visual words and each descriptor has its own codebook. In the final phase, the authors aimed to use a classifier that do not require training before use. Therefore, they used the affinity propagation clustering algorithm defined in [18] for action classification.

The proposed violence detection algorithm was tested on three publicly available datasets, namely the Mind's Eye[19]the Subway [20] and the Vanaheim Metro [21]. It achieved decent detection accuracy on all the datasets. However, due to the use of a complex video descriptor, this action recognition solution is computationaly expensive, so it cannot run in real time on a smart surveillance camera that has limited hardware resources.

Another interesting action recognition algorithm, is proposed in [22]. The algorithm is able to detect six actions in videos, namely walking, standing, bending, lying squatting and sitting. In order to detect those actions, the algorithm uses shape of the human bodies and the AdaBoost algorithm. Its principle of operation is illustrated in figure 9.

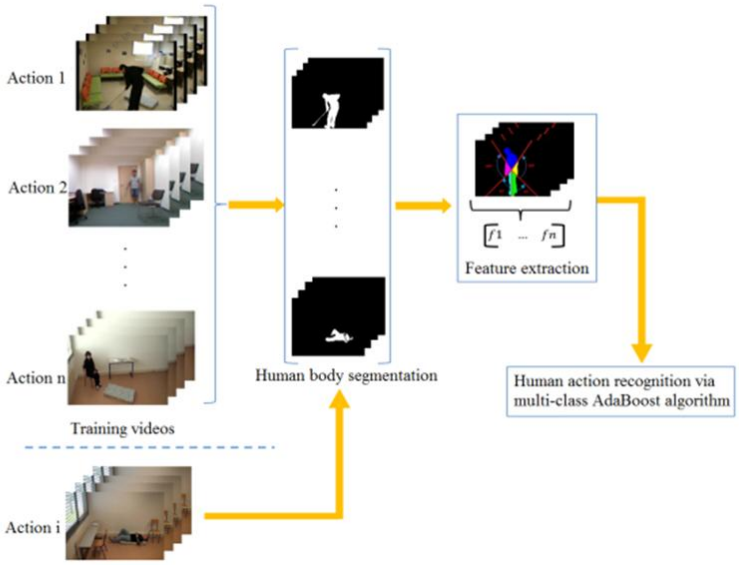


Figure 9 - Illustration of internal operations of the human action recognition algorithm [22]

The human body segmentation stage of the algorithm is responsible for extracting the silhouettes of humans in the video. For this purpose, the authors decided to use a custom background subtraction algorithm for separating the foreground objects from the static background. As was to be expected, the foreground mask generated by this method contains a lot of parasite (noise) pixels that affect the segmentation performance of the proposed algorithm. To remedy this issue, they use morphological operations in the process. Namely, the authors use the erosion and the dilation operations [23] to get rid of many parasite pixels in the foreground mask. Those wrongly classified pixels in the foreground mask are usually caused by the inability of the background subtraction to precisely construct the background model. Due to the efficiency of this noise elimination method, I decided to use it in the algorithms I propose in chapters four and five.

Feature extraction is the next step of the algorithm of [22].The proposed approach extracts the features of humans by dividing their silhouettes into five regions. Such features are simple, invariant to rotation and translation and yet descriptive enough to distinguish among the five human actions targeted by the authors.These features are easy to extract and are very robust. They also do not demant complex operations in their extraction process. So these findings, motivated me to use a similar silouethe based feature in the algorithm I propose in chapter five.

Classification of the actions is achieved by using the AdaBoost algorithm. The idea behind this learning meta-algorithm is to use several simpler classifiers linked in a cascade manner. Therefore, the AdaBoost classification algorithm proposed in [22] consist of a cascade composed of several random tree classifiers. The approach is One-Against-All. Each "weak" classifier is a binary classifier that can classify the data into just two classes. That means that each classifier is able to recognize just one action. If the current action is not the action for which the classifier was instructed, it is passed to the next classifier in the cascade.

This promising approach claims to achieve a decent detection performance on the two of the most well known datasets, namely the fall detection dataset (URFD)[24] and the UMAFD dataset [25]. As its name suggests, the first dataset is concerned with detecting the fall, while the second one is more generic thus containing daily activities. The algorithm succeeded to detect the fall in 96.56% of cases in the URFD dataset. The detection of activities in the UMAFD database follows the same trend. According to the result presented in the paper the overall action detection accuracy for the UMAFD dataset is 93.91%.

The shortcoming of this system is the use of a custom background subtraction algorithm that does not automatically update its background model. Consequently, the proposed action classification algorithm requires fine tuning during operation. Moreover, the algorithm is designed to recognize the actions of a single person,

which makes it unsuitable for the analysis of urban surveillance videos for violence detection.

In [26], the use of a motion template for the recognition of actions in the video is proposed, namely the temporal template. In fact, this is a template matching technique. According to the data provided in the paper, this algorithm does not require a computationally expensive motion extractor. It uses a simple frame difference technique for extracting the person motion. This algorithm inspired me so I used a similar motion feature in my algorithms.

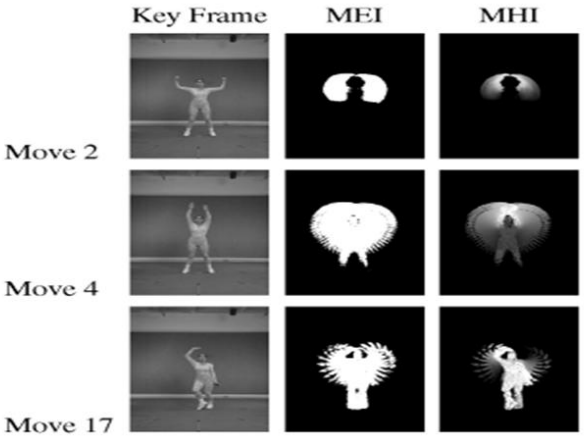


Figure 10 - Comparision of the MHI and MEI descriptors for three types of human actions [26]

The temporal template is nothing more than a vector image that encapsulates the motion of the object in the scene. In this regard the authors use two motion descriptors and a statistical descriptor to construct it. In the figure 10the left column shows the original frames, the middle columnillustrates the Motion Energy Image (MEI) descriptor and the right column illustrates the Motion History Image (MHI) descriptor.

The MEI descriptor is computed by the means of a frame difference technique. It is a cumulative binary motion image that is captured from several consecutive frames. The MHI descriptor is similar to the MEI descriptor, the only difference being that it additionally encodes the direction of motion. For coding it uses the time at which the movement took place. Pixels that have a lighter color represent the motion that took place a short time ago, while pixels darker in color represent the motion that took place some time ago. Therefore, this encoding allows the algorithm to capture the direction of motion.

These descriptors are further processed by the extraction of seven Hu moments algorithm [27] resulting in a statistical descriptor which is actually the temporal

template. Because the authors use a template matching technique, they construct statistical models of the actions. They use training samples that contain several types of actions captured from different angles to generate such temporal template models. Finally, in the inference phase of the algorithm, the temporal template of the action in the scene is compared with the stored action models.

The experimental results conducted on an aerobic data set show us that this solution can successfully recognize the exercises of an aerobics instructor in video.So I think it can also be succesfully used to recognize human actions in videos. However, the approach has some problems too. It is designed to detect actions of just one person in the scene so it will not work if there are two or more persons in the scene. Also it is very sensitive to occlusions so it will fail to detect the action if the person is occluded by another object. These drawbacks make this approach to be unusable in real life scenarios. Therefore, this algorithm is not suitable for a smart surveillance camera that needs to analyze real life scenes.

An improved version of the same template matching approach that I investigated is defined in [28]. The authors of this paper realized that the volumetric space time shape of human action is a more appropriate descriptor for the action recognition algorithm than the 2D shape descriptors used in [26].

Figure 11 - Illustration of space time shapes for three types of actions,namely for jumping-jack, walking and running [28]

Figure 11 is an illustration of such space time shape descriptors for the actions of jumping-jack, walking, and running. We notice that the motions of each action are almost perfectly captured by this simple technique. The space time shape of an action is easily obtained by concatenating the actor’s silhouettes for several consecutive frames.

For modeling actions, the approach uses the shape features like shape dynamics, shape orientation and shape structure. In this sense, the authors use a mathematical equation, namely the solutions to the Poison equation to extract the features of the volumetric space time shape of the actions.

Because the solution is based on templates, it requires a database that contains human action models. So, in the training phase, the authors extract features of the actions that are to be detected. In the detection phase the extracted features of the analyzed scene are compared against these stored models by using the Euclidian distance. A match is found when this distance is below a predetermined threshold.

This method is indeed an improved version of the algorithm proposed in [26] but it has the same limitations. It is designed to recognize only the action of one human and is very dependent on the angle at which the silhouette is captured. These disadvantages make the algorithm unsuitable for analyzing real live scenes.

4.1.2 Solutions that use local features

In this chapter, I am going to contine to describe the computer vision algorithms for detecting action in video, but focus only on those that use local features. Therefore, a computer vision approach for capturing unusual human actions that could be used in a smart surveillance camera to detect human behavior is described in [29]. The authors propose several algorithms that are aimed for improving the safety of truck drivers while they are in a parking lot. An unwanted action such as the act of theft often takes place in a truck parking lot. Usually the driver is attacked by thieves to steal his cargo.

Therefore, the authors designed an interesting handcrafted automated video analysis solution to capture the theft action. The solution includes the following steps:

- preprocessing
- zone based activity learning
- individual action recognition
- group formation

In order to be able to detect suspicious actions that can lead to an attack to truck driver, the algorithm requires knowledge of the number and locations of pedestrians in the scene. To detect the pedestrians, the proposed solutions uses the Gaussian Mixture Model (GMM) background subtraction algorithm defined in [30] and the Fastest Pedestrian Detector in the West (FPDW)[31] algorithm. The GMM is used to extract foreground objects, while the FPDW is used to classify the foreground objects in either pedestrians or non pedestrians. Also, in the preprocessing phase, the proposed algorithm for detecting theft actions uses a Multi Hypothesis Tracker (MHT) [32] to track pedestrians. This approach seems to be very effective in detecting the pedestrians. Inspired by this finding, I decided to use the GMM background subtraction algorithm in the algorithms that I propose in chapters four and five.

The next step, namely zone based activity learning, involves the automatic detection and labeling of the activity zones. These are the zones in the scene where two or more people meet or where a person changes his or her behavior, that is starts running or stops walking. The algorithm inspects tracklets for speed changes in order to detect changes in people's behavior. This is in fact a simple task that can reveal the behavior of the traced persons so I also decided to use it in my algorithm that I propose in the chapter five.To obtain the activity zones, the algorithm further uses the Leader algorithm [33] in order to cluster the selected pedestrians.

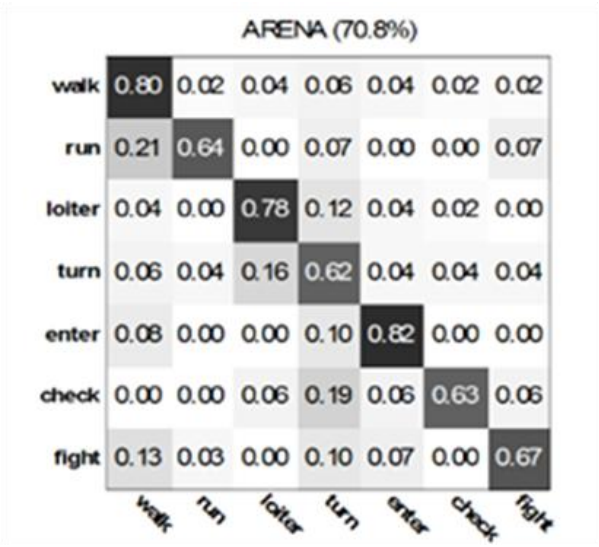


Figure 12 - Confusion matrix of activities detected by the algorithm [29]

Recognition of human actions in the scene is achieved by using the STIP (Spatio-Temporal Interest Points) features[34],the random forest algorithm and the SVM algorithm. The purpose of the STIP features is to select the points of interest of the observed pedestrian in the space-time domain. These extracted points allow the random forest to build the histogram of visual words. Human actions are then detected by the use of histogram and SVM classifier.

The authors considered that the formation of the group is another good indication of an attack scenario. Therefore, they use the K-means algorithm [35], the silhouette of pedestrians and the density measure of group to detect the formation of the group. By combining these features, the algorithm is able to tell when two or more people are standing next to each other thus forming a group.

As the attack on the truck driver by thieves involves a complex behavior, the authors propose to use the activity zones, individual recognition of actions and the

formation of the group to accurately capture the attack action. According to the results published in paper, the algorithm obtained an average accuracy of approximately 70.8% in recognition of seven human actions on the ARENA2013 data set. The confusion matrix given in figure 12shows the scores for each action.

These results are quite good, but there is room for improvement, as many actions are misclassified. The algorithm uses the STIP features that demand complex calculations. This process requires serious computing resources that are not available in a smart surveillance camera device.

Another state of the art approach that could be used in a smart surveillance camera for detecting human behavior in video is presented in [36]. The proposed solution is able to detect only violent actions in videos, such as fights.

The recipe of this algorithm is almost identical to the action detector proposed in [29]. The algorithm is based on the Bag of Visual Words (BoVW) method and it uses the STIP [37] and the MoSIFT [38] feature extractors to detect the interest points of the fight action. The visual word vocabulary is build by the use of a K- means clustering algorithm. Thus , the center of each cluster is a visual word. The last stage of the algorithm is responsible for classification. For this task the authors decided to use the SVM classifier.Thus, in the training phase, the SVM classifier was trained with the attack samples.

Due to the lack of data sets containing fights, the authors built a new video data base called the Hockey dataset. They recorded a hockey game and manually labeled the fight actions. I believe that the selection of a hockey game was made because it contains many fights between players. The detection accuracy of the proposed fight detection algorithm for these videos is around 90%. The algorithm has also been tested on action movie clips that contain fights. According to these data, the algorithm seems to work quite well in detecting fight events. Both STIP and SIFT feature extractors are robust algorithms. The only downside is that they use complex calculations, so the smart surveillance camera can't run them in real time.

The approach proposed in [39] is another interesting solution for capturing human actions in video. The authors propose an algorithm for human violence detection. In this context, they define violence as individual or group fighting in public spaces.

The proposed algorithm introduce a novel descriptor, namely the Violent Flow (ViF). This descriptor is based only on the motion information extracted from the video. It encapsulates the changes of the optical flow magnitude over time. The features extracted by the ViF descriptor are classified by the popular SVM classifier. Of course, before use, the classifier is trained to detect the violent motions. The algorithm is able to classify the scene into violent or non violent. This approach

caught my attention because it uses only motion data and the SVM classifier, so it seemed to be a simple solution that does not require large computing power.

The authors designed three versions of the algorithm. The first use the Lucas-Kanade [40] optical flow to construct the ViF, the second one uses the Horn-Schunch [41], and the third uses the Iterative Reweighted Least Squares (IRLS) [42]. They aimed to compare the performance of the proposed violence detection algorithm based on the optical flow algorithm used. The Lucas-Kanade version of the algorithm obtained 96.5% on the Movies dataset, 74.1% on the Hockey dataset and 69.11% on the Crowded dataset. The Horn-Schunch version obtained 96% on the Movies , 81,3% on the Hockey and 74,8% on the Crowded dataset. Finally the IRLS version scored 100% on the Movies, 77.9% on the Hockey and 69.11% on the Crowded dataset.

After detailed investigation I realized that the solution is quite effective in detecting violent actions, but it actually requires a lot of computational power. Employing the optical flow to extract motion from the video is a dealbreaker for using this solution in a smart surveillance camera. It requires large computing resources that are not available on such a device.

Inspired by the promising result of using motion feature for recognizing actions in video and consequently recognizing the human behavior I explored the solution presented in article [43].The authors designed an algorithm that uses the dense trajectories. Operation of such action detection system is based on analyzing the trajectories of optical flow. In addition to optical flow, the authors include additional features in the algorithm, such as the object appearance descriptor and the localmotion descriptor.

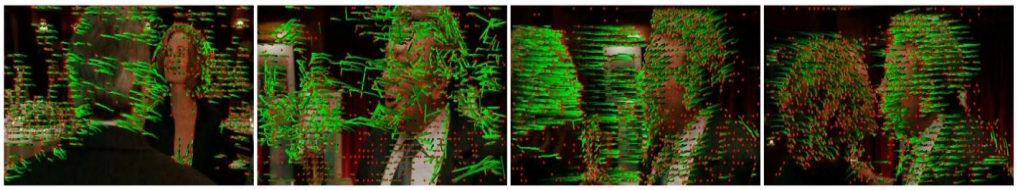


Figure 13 - Illustration of the dense trajectories descriptor [43]

An illustration of dense trajectories is given in figure 13. The green lines in the four frames illustrate the dense trajectories of the upper parts of the human body. According to the theory presented in the paper [43] this descriptor is very effective in capturing motion in video. Even the slightest movements are detected successfully. This finding motivated me to further investigate the proposed approach. Capturing motion in such fine detail theoretically means that all human actions can be recognized by using this motion descriptor.

The block scheme of the proposed algorithm is illustrated in figure 14. In the first phase the algorithm applies the Farneback [44] method to calculate the optical flow over the whole frame. But because the motion of each pixel in the video does not need to be captured, the authors decided to sample the optical flow data. For this purpose they use a grid that is spaced by five pixels. This process is illustrated in figure 14 left. The algorithm is designed to work on multiple spatial scales. Therefore, optical flow extraction and dense sampling processes use the original frame and a few other scaled frames. That's why there are four grids in the figure 14.

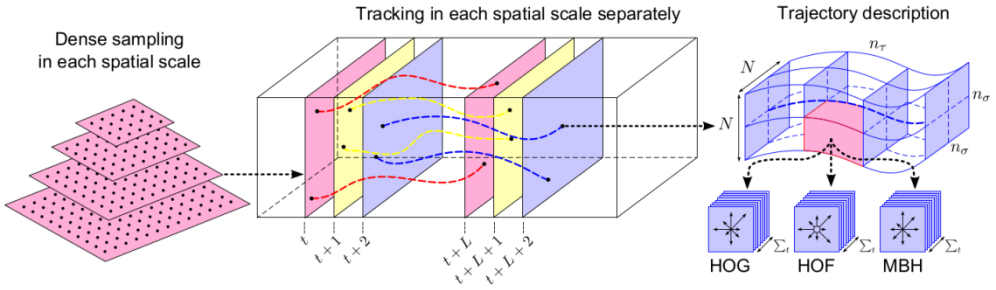


Figure 14 - The process of generating the dense trajectories descriptor [43]

Tracing all the sampled points is not effective for detecting the action. Hence to reduce the complexity of the algorithm the authors decided to remove the sampled points that are extracted from the homogeneous areas in the scene. The optical optical flow in these areas should be none, but due to the camera noise, they contain small amount of optical flow motion. These areas are not relevant for recognizing the actions, as they do not contain any human movements. Therefore, the authors eliminate these points by using the algorithm proposed by Shi and Tomasi in [45].This method of removing the motion data that is not relevant for the action recognition attracted my attention so I used a similar approach in the solution discussed in chapter six.

After sampling and removal of the irrelevant motion data , the remaining motion features generated by the optical flow algorithm are tracked throughout the video. For obtaining better detection accuracy, the algorithm limits the length of the trajectories. It uses a time window of fifteen frames. So the dense trajectories are a result of tracking optical flow features along this time window.

At the end of time frame the algorithm computes the shape of each trajectory. The shape is a single normalized vector which is calculated by summing all the components of the trajectory. Before using dense trajectories, the algorithm removes erroneous trajectories, such as trajectories with large displacements. While

researching this phase of the algorithm, I realized that tracking the motion of objects and analyzing the shape of their trajectory can reveal us usefully information that I can use to detect their behavior.

To further improve detection accuracy the proposed algorithm computes the HOG (histogram of oriented gradients), HOF (histogram of oriented flow) and MBH (motion boundary histograms) descriptors. The idea of this approach is to use the appearance and the local motion, along with the dense trajectories of the object in the process of detecting the action. The HOG captures the appearance of object and the HOF and MBH capture the local motion. These descriptors are calculated in each of the trajectory point as illustrated in figure 14 right.

The last phase of the algorithm is responsible for detecting the action. It receives the trajectories, HOG, HOF and MBH descriptors and outputs the action label . The classifier is based on the bag of features approach. For each descriptor the algorithm constructs a codebook by using the k means clustering algorithm. In the classification phase the algorithm uses the SVM classifier to classify the action.

An extensive evaluation of the proposed action detection algorithm was performed on nine action datasets. The best accuracy is 98.4% on the KTH dataset and the worst is 58.2% on the Hollywood2 dataset. Despite of the achieved results the performance of the proposed algorithm dependents on the quality of the optical flow algorithm used. The available methods are far from beeing perfect and often the algorithm can behave poorly. Another limitation is the demand of high computational power for computing the optical flow. This was a deal breaker because the algorithm cannot run in real time on a smart surveillance camera that has limited hardware resources.

A more advanced video analysis solution for capturing violent actions in video presented in [46] uses a complex motion descriptor. It is an interesting approach to detecting violence in crowded and non-crowded scenes. The proposed Bag-of-Words (BoW) framework is based on the motion of the objects in video. It uses the Dense Trajectories (DT) [47] and the MPEG flow motion (MF) [48] descriptors for estimating the object motions.

The motion of the objects in the video is extracted by the DT and the MF algorithms. The DT descriptor is a complex feature extractor made of Histogram of Oriented Gradient HOG, Histogram of Oriented Flow HOF [49] and Motion Boundary Histogram MBH. By combining several feature extractors the DT is able to successfully extract the appearance, the shape of motion trajectories and the motion boundaries of the objects in the scene. This novel motion feature promises a high detection accuracy of actions in videos because it includes two motion descriptors. Motivated by this finding I analyzed more deeply the proposed approach.

Although the MF and DT are descriptors of the same type, they both extract the object motions, the authors use them both in the feature extraction process. The MF descriptor extracts additional motion information thus it improves the overall detection performance of the proposed violence detection algorithm. The MF uses the video demux to extract the motion so basically it does not require any additional processing except of video decoding. This feature of MF means that we can extract the motion of objects at almost no computational cost. Therefore, the MF is a very efficent motion extraction algorithm that can be run on low resource hardware. So, after a few experiments I decided to exploit the MF in the solution that I propose in chapter six.

The output of both descriptors used by the proposed algorithm is a high dimensional vector. The DT provides 426 dimensional vector and the MF gives 396 dimension vector. This huge amount of data makes it hard the process of learning the dictionary. To face this problem the authors use the Principal Component Analysis (PCA) for reducing the vectors dimension. However, the reduced data is still large, so learning the dictionary is not an easy task. The next phase of the algorithm involves the feature encoding. The reduced feature vectors processed by the PCA are encoded by the Fisher Vector FV [50] algorithm. In the final phase these features are classified as violence or normal by the use of SVM classifier.

Certainly, to function properly the algorithm requires a training phase. The codebook of the BoW model is constructed based on the Gaussian Mixture Model (GMM), the K means ++ [51] algorithm and a set of training videos.

The experimental section of the paper convinced me that the proposed approach performs quite well. It was tested on two datasets. The best detection accuracy on the Hockey Fight dataset [52] is 95.8% and the best detection accuracy on the Crowd Violence dataset [53] is 95.11%. After a complete analysis of this appealing solution, I conclude that it may be suitable for use in a system that has a high computing power. The reason why the solution requires a high computing power is due to the algorithm for extracting dense trajectories. Therefore, the proposed violence detection algorithm cannot be run in real time by hardware with limited resources, such as a smart surveillance camera.

4.2 Deep learning video analysis solutions for action recognition

In this section I investigate the video analysis algorithms that are based on deep learning approach. The first part of this section contains a short theoretical background, while the second part explores several state of the art computer vision algorithms that use deep learning for detecting action in video.

4.2.1 Brief history

The next level in image and action recognition are the deep convolutional neural networks. These types of network have been invented long time ago but were not extensively used until now because of lack of computational power, large databases and efficient training schedules.

Among the first pioneers who used the neural network to recognize patterns in images is the K. Fukushima in the article [54] dated from the 80's. The name of network used in this article is "Neocognition". It was inspired by the human visual system. Hence the first layer of the network is made of cells that correspond to the photo receptors of the human retina. The following layers are made of S-cells and C-cells. The S-cells layers mimic the operation of the primary visual cortex. Features, such as lines or edges are extracted by this layer. Layers that are made of C-cells are based on the operation of the human visual cortex. In these layers the network extracts more complex features such as graphical patterns. To recognize the patterns in the image, the network requires training before use.

The deep convolutional neural networks that proved to be so successful in image recognition in today's VA applications, use the identical principle. They consist of several linked layers which are responsible for extracting image features. The convolutional neural networks belong to the big neural networks "family". It is a special network designed to recognize patterns in image. In the flowing section the operation of this type of network is briefly reviewed.

4.2.2 Internal operation

The operation principle of the deep CNN is similar with the operation of a traditional deep neural network (NN). Deep signifies the network has more than just three layers. They both are composed of interconnected neurons whose links have weights, which multiply the incoming signal. The main difference between these two networks is the way neurons are connected between layers, i.e. the architecture of the network. In the case of a deep NN, one neuron in a hidden layer is connected to all the neurons in the previous hidden layer. This is not true for the deep CNN. A neuron from a hidden layer is connected only to a group of few neurons from the previous hidden layer. Each such group of neurons receives information from a specific part of the image. As a result, the neuron is activated and enforced to extract only local information, i.e. local image features, in the case of image inputs. A major benefit of the convolutional architecture is that, for the same number of neurons, the number of weights to be learned is several orders of magnitude smaller than the number of weights for a fully connected architecture. Consequently, considerably less data is needed to avoid overfitting problems at

training. Yet, the model is complex enough to capture vital information for recognition.

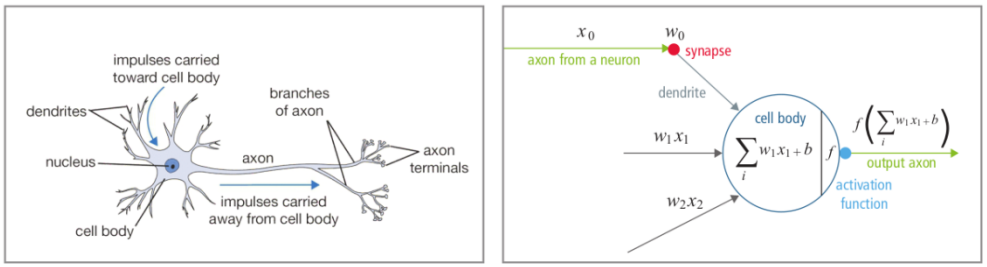
The neuron used in neural networks can be thought of as a mathematical model of the biological neuron. The illustration in figure 15 shows the biological neuron (left) and its corresponding mathematical model (right). Its principle of operation is quite simple. If viewed as a black box, it has inputs named dendrites and an output, named axon.

The signals received thru the dendrites (or dendrons) are firstly multiplied by weights, namely the synaptic strengths of the cell, and then forwarded to the cell body (the black box). This means that the signal received thru a dendrite may or may be not entirely used in the final result, because of the multiplication constant. The dendrites of a neuron are actually the output signals of the axons of previous neuron cells. So, the usage of the outputs of the previous neurons in the current cell is influenced by the multiplication constants. These constants are the learnable parameters of the network.

After multiplication with the synaptic strengths, the dendrite signals are summed up by the neuron cell connected to them, which also adds a bias value to the sum. The interesting part is that this sum is not forwarded directly to the axon, instead the sum is thresholded according to a certain rule (embedded in the activation function). So, if the value of the sum exceeds a certain threshold condition, the cell will be activated, thus sending an impulse through its axon. A mathematical model of a neuron illustrated in figure 15.bis described by theformula (1):

$$y = f(\sum w_i * x_i + b) \tag{1}$$

The y is the output, x_i are the inputs, the w_i are the weights, b is the bias and f is the activation function.



a) b)

Figure 15 - a) biological neuron ; b) mathematical model of the neuron [55]

The activation function f is a nonlinear function whose purpose is to allow the network to learn complex data. Without the nonlinearity, a convolutional network

with several layers would remain linear and do just convolutions. Each neuron in the network has its own activation function.

In practice, many types of activation functions are used. They accept one real number and perform a certain mathematical operation on it. Among the most popular are:

- the sigmoid
- the tanh
- the Rectified Linear Unit (ReLU)

The graph of a sigmoid function is illustrated in figure 16. Its mathematical formula is given by equation (2).

$$\sigma(x) = 1/(1 + e^{-x}) \tag{2}$$

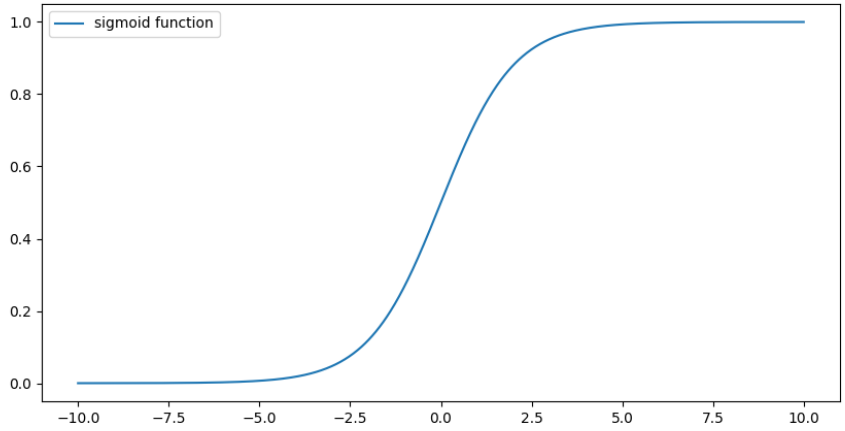


Figure 16 - The sigmoid activation function

The function limits the input number to either 0 or 1. The large negative numbers are transformed to 0 and the large positive numbers are transformed to 1. The main drawback of using this activation function in neural networks is that when saturated it "kills" the gradients so the network can barely learn in this situation. The function gets saturated when its output is at is limits that is 0 or 1. Another drawback of the sigmoid activation function is that its output is not zero centered. This negatively affects the gradient descent learning algorithm during the learning phase of the network.

The second most used activation function in neural networks is the tanh function. The illustration of this function is found in figure 17.

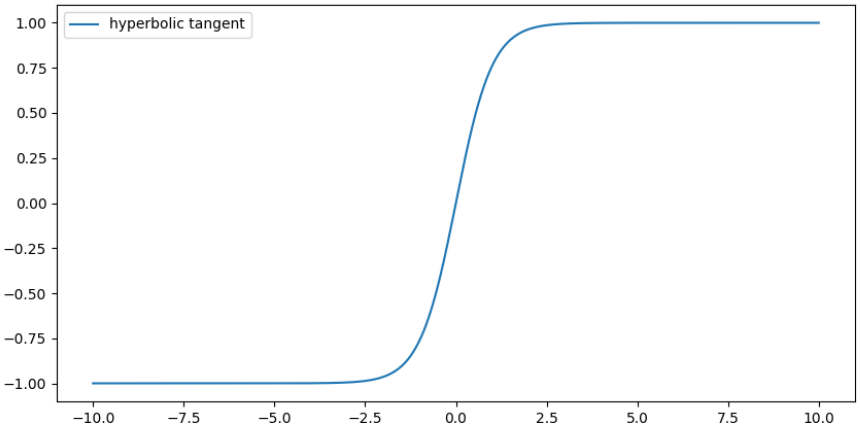


Figure 17 - The tanh activation function

As opposed to a sigmoid function the tanh function is zero centered. It limits the output values to either -1 or 1. In a similar way as the sigmoid function, the tanh function transforms the large positive numbers to 1. In the case of large negative number the function limits it to -1. The mathematical formula of this function is given in equation (3).

$$\tanh(x) = 2\sigma(2x) - 1 \tag{3}$$

Because it is zero centered the tanh function is more frequently used than the sigmoid function in a neural network. Its disadvantage is that it can saturate and kill gradients, just like the sigmoid function.

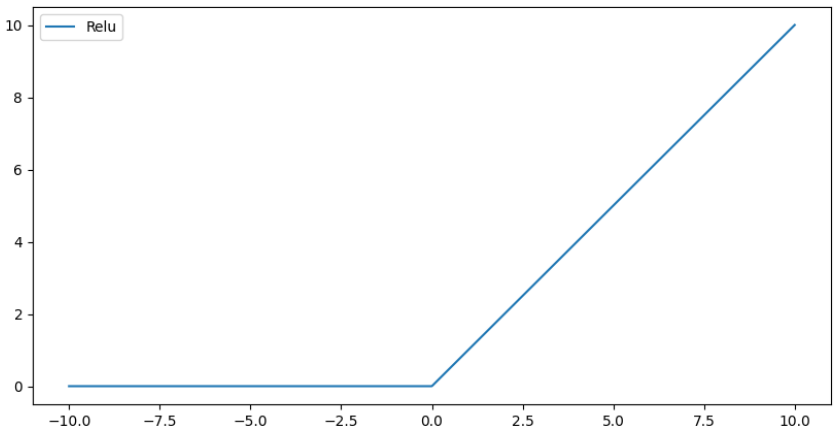


Figure 18 - The rectifier linear unit activation function

The most popular type of activation function is the rectifier linear unit (ReLU) function. Unlike the sigmoid and the tanh functions the ReLU "cuts" the negative numbers as illustrated in the figure 18. That is, it transforms any negative number to zero. It does not affect the positive numbers. The mathematical formula of the ReLU function is given in the equation (4).

$$f(x) = \max(0, x) \tag{4}$$

The strong points of the ReLU activation function are that it accelerates the training process and the function is not computationally expensive [56]. Despite of its benefits the function has its weakness that is it in certain situations it may make the neuron die (for negative activation values). This means that the weights of a neuron update during training may be suppressed andthat neuron may never activate.

A variant of the ReLu function meant to solve the dying issue is the leaky ReLu function. This modified function does not "cut" completely the negative values of the input as ReLu does. Instead it has small slope *a* on the negative inputs branch, thus the negative input values never become zero after processing.

The mathematical formula of the leaky ReLU is given by the equation (5):

$$f(x) = \begin{cases} \alpha x, x < 0 \\ x, x \geq 0 \end{cases} ; \propto = 0.01 \tag{5}$$

The convolutional layers are operating a feature extraction process on the image. To recognize objects or perform other tasks, like object detection (i.e. recognition plus localization), tow other kind of layers are needed: pooling layers and fully connected layers. The block scheme of a typical deep convolutional neural network designed to operate on images is shown in figure 19.

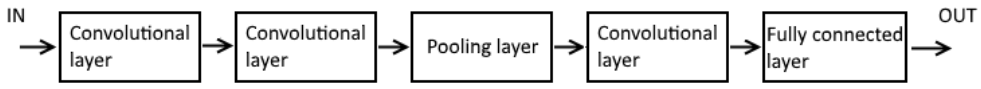


Figure 19 - Typical CNN architecture

The deep CNN analyses pixel by pixel so the input is raw image whilst the output is a vector of classification probabilities. The network consists of:

- convolution layers
- pooling layers
- and fully connected layers

4.2.2.1 The convolution layer

The convolution layers are the main components of the network. Each convolution layer is responsible for extracting image features. Therefore, layers that are close to the input extract basic features of the image, such as corners, edges, lines, while deeper layers combine the basic features detected, to extract more complex features, such as different types of shapes. In figure 20 an illustration of the features extracted at each layer of a neural network that is composed of eight convolution layers is given.

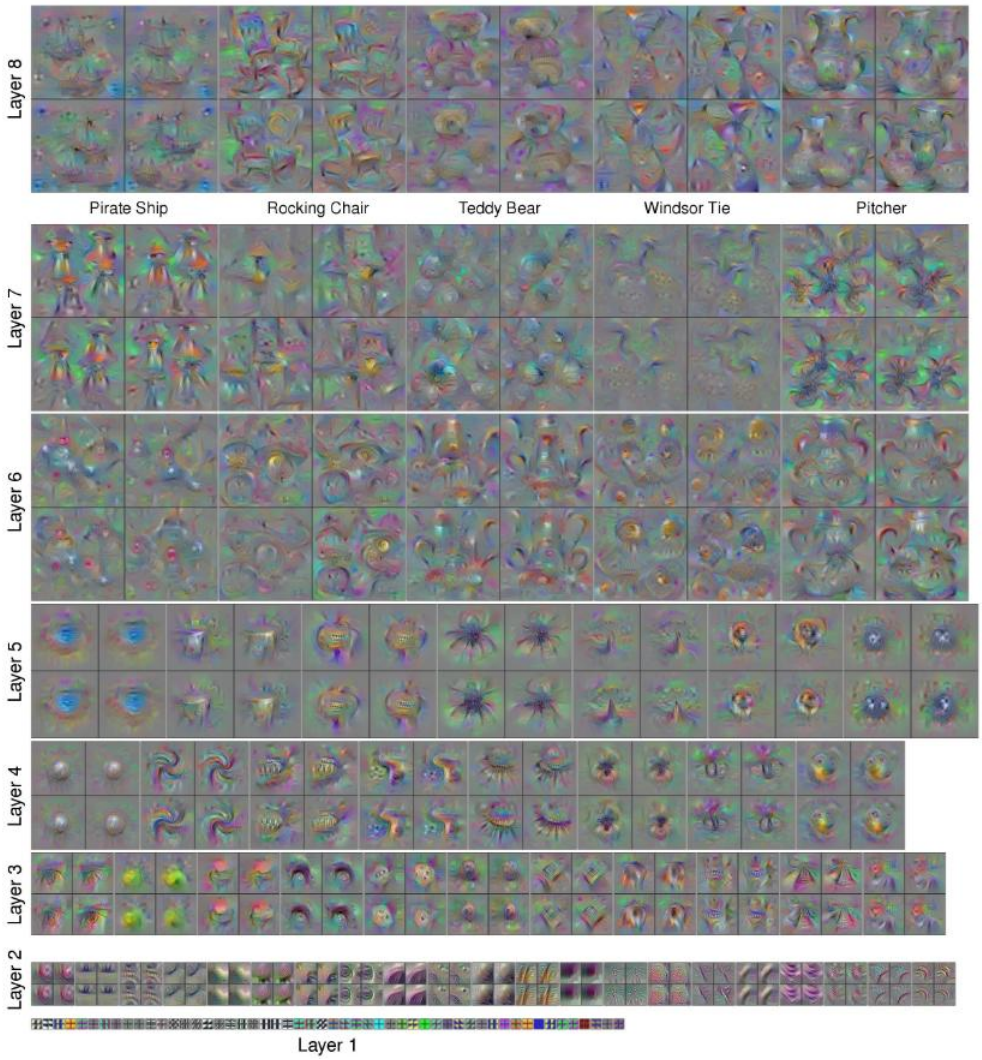


Figure 20 - Features at each layer of the CNN [57]

The convolution layers perform extraction of features from the input image or previous layers by using learnable filters, defined by the neuron weights. These convolution filters are actually simple processing units. They are defined by small sized matrices called the kernels and they perform multiplications of the inputs with corresponding weights, add all results of multiplications, and pass the sum to the nonlinear activation function units.

Unlike with conventional convolution filters used in signal processing, each convolution filter has also a parameter called stride, meant to downsample the output of a convolution layer. The stride parameter tells the filter how many pixels to slide the kernel on the image to get the result for the next output. In conventional convolution, the kernel is translated only one data sample at a time, so it will be centered on each sample at some step and the output will have the essentially the same dimension as the input. Small differences may exist, depending on how the data at the image border, which miss some neighbors is handled. This problem is discussed later on.

The size of a filter kernel is very small compared to the image size. Usually, it is 3x3 or 5x5. An illustration of the operation of a filter is given in the figure 21. The filter slides through the input image (left) which has the size 32x32x3, say from left to right with a stride one, until it processes all the pixels. Upon completion, it outputs the image features namely the activation map in form of a matrix.

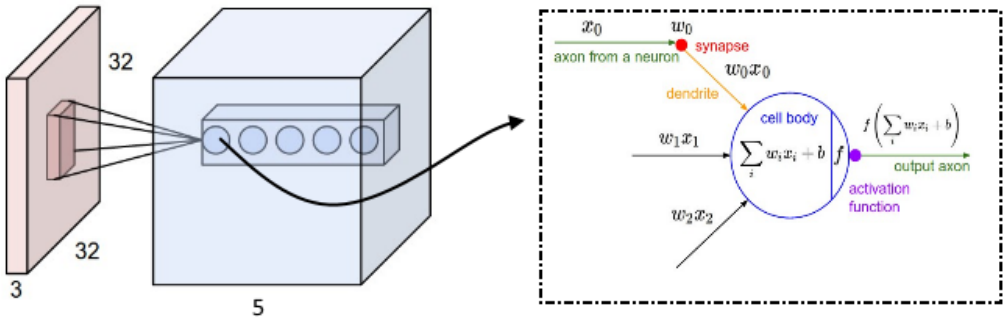


Figure 21 - Structure of a CNN layer. Input image and filter window left in pink color. Activation maps(AM) in blue. Neuron cell is in the right. [55]

In a serial implementation, ateach iteration the filter computes the dot product of the kernel matrix and the sampled matrix of the image, adds a bias and applies an activation function. The sampled matrix consists of image pixels contained in the filter at one iteration. In other words, the filter makes the dot product between the kernel and the pixels in the image with which it overlaps. In the next iteration, the filter slides by the number of pixels indicated by stride parameter and repeats the same mathematical operations.

The input is not necessarily the input image, it can be the activation maps of the previous layer. If the input image has multiple channels (like R,G,B), convolution is computed in a similar manner. The convolution kernel needs to define the weights for all channels. Unlike in image or signal processing, the number of output channels is not necessarily the same as the input channels. To generate a desired numbers of output channels, we simply use the desired number of convolution kernels and perform convolution with each of them.

As for the analogy with the brain, a convolution filter consists of a limited number of neuronal cells. Each neuron is connected to only a small number of pixels in the image. Therefore, each one processes just a small area of the image.

The input size or number of weights of a neuron is equal to the total number of pixels with which the neuron is connected times the number of channels. In figure 21 the input size is [5x5x3] because the filter size is 5x5 and the image has three channels, hence the kernel of each neuron has 75 weights. Those weights are learned during training and shared among the neurons that belong to the same filter.

The pixels connected to a specific neuron are multiplied by the neuron weights, then summed all together. A bias is added to the result and then an activation function is applied to the sum. Each neuron performs the same type of operations, i.e. convolution using its learned weights and the nonlinear activation function computes exactly one value of the activation map for each convolution kernel. In figure 21 there are five convolution filters in one layer, so the layer receives an RGB image and produces five activation maps.

A shortcoming of the convolution operation is that it reduces the spatial size of the input, if the operator is constrained to keep its window within the input data matrix. The size of the activation map is not equal with the size of the input image, with this constraint. To maintain the input and output data support equals, the input needs to be zero padded. Usually, before applying the convolution filter of size 3x3, having the stride parameter set to one, the input image is zero padded by one pixel along its borders.

Other image padding solutions exist, like using nearest neighbors. For a 5x5 filter size, two pixel wide image padding is used and so on. This way the size of the input remains unchanged after processing, which is the most frequently used solution. Nevertheless, some networks use the convolution layers to shrink or extend the input shape. If the input data is not padded at all, the convolution operation shrinks the input shape. Conversely, if the input data is padded, the output extends or retains its original shape.

4.2.2.2 The pooling layer

The convolution layer captures features from the image. Even the positions of features in the image are implicitly encoded in the activation map. Convolution filters produce strong activations where the convolution kernel matches the input data. This happens whenever the kernel location is in a close neighborhood of a matching pattern. It is the task of the pooling layer to keep only the strong activations while downsampling the layer connected at its input. It removes the unnecessary data by reducing the size of the activation maps. The function of the pooling layer is similar with the function of down sampling used in signal processing.

Like convolution, the pooling operator has a kernel that processes all the pixels within the support window. However, the kernel is not evaluated at all input locations: it down-samples the input data. Windows of the max pool operator may overlap or not. A frequent configuration of the pooling layer used in the CNN is a kernel of size 2x2 and a stride of 2. This setup reduces the input size by 2.

At each iteration, the pixels encompassed by the kernel are processed by one of the following functions:

- max pooling
- average pooling

As its name suggests, the average pooling calculates the average of the pixel values within each kernel window. On the contrary, if using the max pooling, the maximum value pixel is chosen as the result while the remaining pixels of the kernel window are ignored. So the filter usesseveral inputvalues and outputs just one value. An example of a max pooling operation is illustrated in the figure 22. The max pooling function is most commonly oneused in the pooling layers of the CNN.

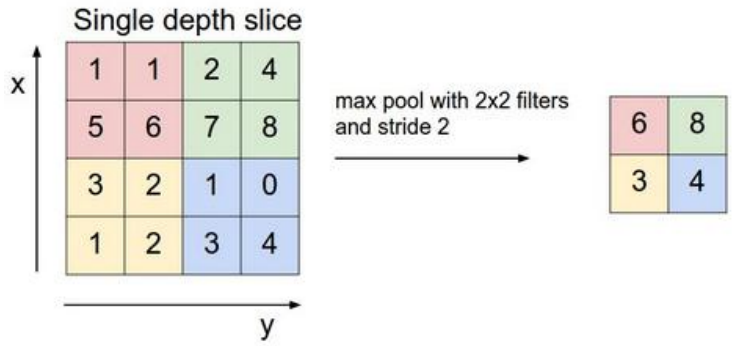


Figure 22 - The max pooling operation [55]

While the first few layers of a deep CNN capture local image features, the next levels discover more complex structures made up by groups of local features, like grouping edges into lines or circles. As the size in the original image layer covered by such a group gets larger and larger, the down-sampled image can capture more complex and relevant information for recognition with less neurons, which is beneficial for keeping the model size as low as possible.

4.2.2.3 Fully connected layer

One layer of a CNN consists of several convolution filters, working in parallel, thus each layer extracts several image features at once. The resulting activation maps are further supplied to the next convolution layer. In between consecutive convolution layers there is usually a pooling layer. And at the end of the network there is a fully connected layer. The role of this last layer is to perform the classification of the high-level features. In a fully connected layer, each neuron is connected to all the inputs. As with convolution layers, each connection includes a weight, to be learned.

There is a direct link between the number of neurons in the final fully connected layer and the number of classes that the network has been trained to recognize. The number of neurons is equal to the number of classes, so the output of a neuron can be interpreted as the probability for a class. In fact, to add up to unity like probabilities, the network outputs are generated by the SoftMax activation function and not directly by the neuron outputs. The SoftMax function transforms the real numbers computed by the neurons into numbers that can be interpreted as probabilities. To this end, each neuron's output is rescaled according to theequation (6).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{6}$$

The integer K represents the number of classes, the variables Zi are the input elements for the SoftMax function and the e^{zi} is the exponential function on neuron output variables. The SoftMax transforms the real numbers computed by the neurons, that can be either positive or negative, into positive probabilities. This activation function is used only in the fully connected layer of the CNN, while the other layers in the network use the sigmoid or the tanh or the ReLu activation function. The only exception to this rule is the pooling layer. It has no activation function.

In summary, the CNN is a special type of artificial neural network that was inspired by the human visual system. The network is capable to classify images according to their content. Therefore, it accepts an image and produces a vector of class probabilities. The class with the highest probability is most likely to be the actual class of the processed image. The structure of a deep CNN consists of several

convolution layers, several pooling layers and a fully connected layer. All are connected in a series architecture. The most important part of the learned model is encoded by the weights and biases of the convolution layers. They act like filters that extract image features, such as edges, lines and complex shapes. The pooling layer removes the unnecessary data, and the fully connected layer performs the final classification. Similar to convolution layers, the fully connected layer also has weights and biases that it learns during training.

4.2.2.4 Supervised training

The CNN network can be trained in a supervised, unsupervised or a semi supervised manner. It would be ideal if the network could learn on its own, but that way of learning does not give very good results. The best way to train a CNN, when enough data is available, is by supervised training. This training method gives the best results, but it requires the involvement of people in the process, because it requires a huge amount of labeled data.

The purpose of the training phase is to set up the learnable parameters of the CNN. These parameters are the weights and biases of neurons. They need to be changed accordingly to allow the network to correctly classify the input image. When creating the network, the weights and biases are chosen at random, so that the network is not able to extract the correct image features and produce correct classifications.

The training process uses :

- the cost function that aims to tell how far the network's prediction is from the ground truth
- and the back-propagation algorithm whose purpose is to update the weights and biases

A supervised back-propagation training cycle consist of a forward and backward pass of data throughout the network. In forward pass, the back-propagation algorithm finds out how far is the network prediction of a single input image from the ground truth. It uses the cost (or loss) function for this purpose. In backward pass the algorithm tries to reduce the cost function by adjusting the weights and biases for the same input image. This process is iterated many times during training, with different input data, according to a schedule, in a quest for the global minimum of the loss. Figure 23 illustrates the value of the cost function during training.

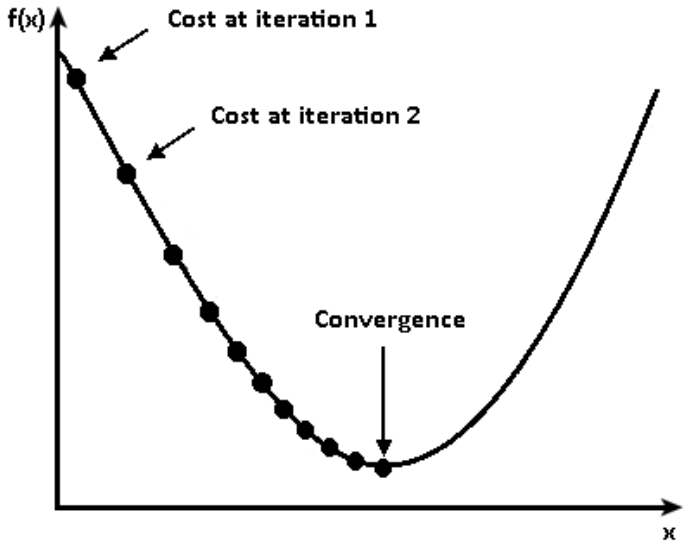


Figure 23 - The cost function during training [58]

The back-propagation process uses the gradient descent algorithm for updating the weights and biases and finally finding the global or local minima of the cost function. It also defines a learning rate parameter whose purpose is to determine the speed of change of the trainable parameters.

The most popular cost function that is used for training a convolutional neural network is the cross entropy function. It comes from information theory and is designed to measure the difference between two probability distributions.The matemathical formula of a cross entropy function isgiven in the equation (7).

$$D(p,q) = -\sum p_i \log(q_i) \quad (7)$$

where q_i stands for the predicted distribution by the network and the $p_i$represent the ground truth distribution.

4.2.3 Neural network based computer vision algorithms for detecting action in video

In this section I investigate several state of the art computer vision algorithms that use the deep leaning approach. I aim to select the best deep learning architecture for action recognition and, at the same time, the one that does not require intensive computing resources. Hence, in the following paragraphs I focus on solutions that use various CNN architectures for recognizing actions in video.

In the work proposed in[59], the authors propose the use of handcrafted features and Convolutional Neural Network(CNN) for recognizing actions in the video. To accomplish this task, the approach uses the appearance and motion of objects in the scene. The Block diagram in the figure 24 illustrates the proposed concept. There are two CNN's involved in the process of action detection. The purpose of the first network is to analyze the spatial domain, and the second network is used to analyze the temporal domain. These two networks are complementary to each other, so that by combining them in the algorithm, the actions in the video can be captured with high accuracy.

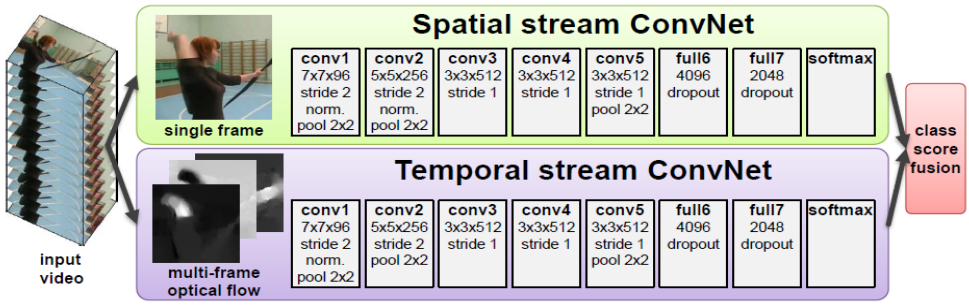


Figure 24 - Two-stream architecture for video classification [59]

The spatial stream ConvNet is a network trained for recognizing actions from still images. Some of the actions are closely related to the object appearance, hence in these cases, the spatial stream ConvNet can recognize the actions on its own. However, not all actions can be recognized in this way. Therefore, in order to allow the recognition of such actions, the authors decided to include the temporal stream ConvNet into the algorithm.

The temporal stream ConvNet is based on the stacked optical flow data. That is, the image frames are processed before being supplied to the network. In this preprocessing stage the algorithm computes the dense optical flow for each pair of frames. This optical flow data is stacked for several frames in order to capture the evolution in the time domain thus resulting an optical flow volume. Because the network is designed to operate on matrices, the optical flow vectors are firstly decomposed into horizontal and vertical matrix components and then stacked. The input of the spatial stream ConvNet are the frames while the input of the temporal ConvNet consists of the stacked horizontal and vertical components of the optical flow.

The configuration of the CNN is illustrated in figure 24. It is a CNN-M-2048 architecture shared from the article of [60]. It consists of five convolutional layers, two fully connected layers and one SoftMax layer. The spatial ConvNet configuration differs from the temporal ConvNet configuration only by one layer. Specifically, the

latter one does not include the normalization layer in the conv2 layer of the network.

In training phase the network use the stochastic gradient descent algorithm to update its weights. Before supplying them to the network, the frames are subjected to the augmentation operations, such as random flip, random crop and RBG jittering.

There are two versions of two stream ConvNet's presented in the article. The first merges the results at the last layer by averaging, thus obtaining an accuracy of 86.9% on the UCF-101 dataset and an accuracy of 58.0% on the HMDB-51 dataset. The second version uses the SVM classifier for merging and thus achieves an average accuracy for UCF-101 of 88.0% and for HMDB-51 of 59.4%.The mean detection accuracy is slightly better when using the SVM classifier.

Using two CNN's for recognizing actions requires a huge amount of training data. The solution proposed also requires a motion descriptor that is based on the optical flow.
These requirements make this approach unsuitable for use on an intelligent surveillance camera, mainly due to the high computational requirements of the optical flow algorithm.

My next investigation that seemed to match what I was looking for was the approach described in [61]. The authors propose two methods for classifying videos according to their content. The first one is based on the temporal feature pooling, while the second method implies the use of Long Short-Term Memory (LSTM) cells in the final layer of the network.

In order to include the time domain in the feature extraction process, the authors propose to process several video frames simultaneously. The interesting fact is that each frame is processed by a separate CNN. Therefore, the number of CNN networks of the proposed architecture is equal to the number of stacked frames. Both methods use either the AlexNet [62] or the GoogLeNet [63] CNN to process individual frames.Along with the raw input frames, the approach also uses optical flow [64] data. The authors used the two stream concept like in[59] for both methods.

The idea of the first method is to add a max pooling layer to the network in order to combine the image features provided by the CNN's. In this regard the authors defined five temporal feature pooling architectures:

- conv pooling
- late pooling
- slow pooling

- local pooling
- and time domain convolution

The conv pooling carries out a max pooling operation over the last layers of the CNN. Similarly, late pooling performs the max pooling operation on the outputs of the two fully connected layers that are added on top each CNN. The slow pooling is a combination of conv pooling and late pooling. It combines hierarchically image features provided by the CNN, that is it applies max pooling over output of several layers of the CNN and then caries out a max pooling over the outputs of the fully connected layers. The local pooling performs the max pooling operation over output of several layers of the CNN. And the last one, the time domain convolution uses an extra time domain convolutional layer that is placed in between the output layer of the CNN and the max pooling layer.

The second method uses the same principle of processing the frames. It has a CNN for each input frame but instead of pooling layer it uses the Long Short-Term Memory (LSTM) cells in its final layer. Based on the experimental data the authors found out that five stacked LSTM cells provide the best detection accuracy. Therefore, the output of each CNN is passed thru five LSTM cells until it reaches the softmax layer.

The proposed methods have a fairly impressive detection accuracy. On the Sports 1 million dataset the best accuracy is 73.1% and on the UCF-101 the best accuracy is 88.6%. The downside of this approach is the use of several CNNs and the use of the optical flow motion descriptor. These two operations require a huge amount of processing power. Therefore, the proposed methods cannot be used in a smart surveillance camera that has limited resources.

Another interesting solution for action recognition is defined in [65]. The authors use a slightly different approach than the one proposed in [59]. The architecture of the network is the same as in [59]. The difference is in the algorithm used to generate motion data. This improvement aims to speed up the inference process, thus allowing the solution to work in real time.

Because the calculation of the optical flow is expensive, the authors decided to use motion vectors generated by block matching in the video encoding process instead of the methods proposed by authors in [66] and [67]. The Farneback's algorithm [68] for extracting the optical flow takes 360 ms per frame and the Brox's flow algorithm [69] needs 60ms to process a frame. These latencies do not allow to run the algorithm in real time.

Motion vectors are generated and used in the most video encoding processes and can therefore be obtained at no computational cost while decoding the video frame. The motion vectors feature is similar to the optical flow feature, although such

vectors estimate motions for blocks of image pixels (typically 16x16 pixels). They both capture the motion in video. So the authors decided to feed the motion vectors instead of the optical flow into the temporal ConvNet.

To gain a higher detection accuracy the approach uses several transfer learning techniques. The scope of this training method is to train at first the temporal ConvNet with the optical flow data, to learn finer motion information, and then, to fine tune it with the motion vector data.

Although this approach proposes an algorithm that claims to have state of the art detection accuracy on the UCF101, HMDB51 and THUMOS14 datasets, it requires two convolution networks to run in parallel. This requirement makes this approach unusable on a smart surveillance camera, as it requires a huge amount of computing power needed to run two neural networks in parallel.

Another solution for action recognition I have explored proposes the use of a novel video descriptor namely the trajectory-pooled deep-convolutional descriptor (TDD) [70]. The TDD is a result of combining the handcrafted features described in [71] with the feature maps generated by the ConvNets [59].

The two features that are used to make the TDD descriptor are:

- the optical flow trajectories
- the feature maps

Trajectories are extracted using improved trajectory algorithm [70], which is an extended version of the dense trajectory algorithm [72]. The extended version is able to extract the optical flow trajectories of videos that are captured with either a static camera or a moving camera. While in contrast, the dense trajectory algorithm only works for videos captured with static cameras.

The feature maps in [70]are generated using a slightly modified version of the two stream ConvNets proposed in [59]. Since the original version offers probabilities and not feature maps at its output, the authors decided to remove the last layers of the network, thus allowing the network to provide feature maps instead of probabilities at its output. As a result, the modified version is a ConvNet without the layers five, six, seven and eight. Another modification of the network is the addition of a zero padding operation before input of each conv layer. This operation preserves the shape of the data.

Once the trajectories and feature maps are calculated, the TDD is computed by the trajectory-pooling method. That is, the trajectory trackletsare used as a guide for summing the feature maps. The TDD descriptor is further encoded by the Fisher

vector algorithm [73] and then an SVM classifier is employed to classify the encoded data.

By using the proposed approach the authors succeed to obtain the highest detection accuracy on the UCF101 and HMDB51 action dataset compared to the state of the art solutions [74][59]. Despite the good results, this approach consumes even more resources than the approach defined in [59]. The use of improved trajectories and dense optical flow is a real bottleneck to implementing this solution on low resources hardware, as it requires high processing power.

At this point, I have realized that solutions that use two or more CNN to detect actions in video are not suitable for use in a smart surveillance camera because they require high computational resources. Therefore I decided to explore a more simpler versions like the one presented in the paper [75]. It is an extended version of the classic CNN meant to include the time domain. This type of network, also known as 3D CNN in the literature, is among the first attempts to use the CNN to recognize actions in video.

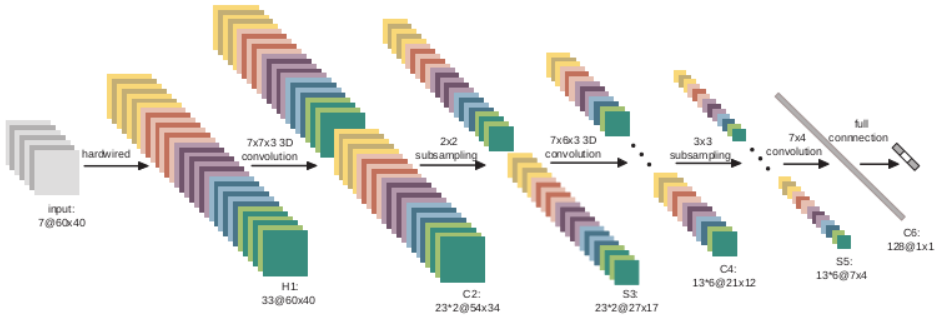


Figure 25 - 3D convolutional neural network architecture for the recognition of human actions in video. The network is composed of one handwired layer, three convolution layers, two subsampling layers and one fully connected layer [75]

The idea behind this approach is to use a stack of multiple video frames in order to capture motion information from the video. In the figure 25 is an illustration of the 3D CNN. An interesting novelty of this approach is the use of a hardwired layer to ease the learning of actions. The hardwired layer has no learn-able parameters. Its weighs and biases are manually established during the initialization phase of the algorithm. The purpose of such an approach is to encode prior knowledge on features into the algorithm. Therefore, the hardwired layer is designed to extract handcrafted features such as gradient, optical flow and gray channel from the video. The following layers consist of classic convolution layers, pooling layers and a fully connected layer at the end of the network.

The reported results of the proposed 3D convolutions reach an average performance of 71% on the TRECVID [76] and KTH dataset. Which is not very impressive for

detecting actions in video. Such low performance may be due to inefficient coding of the features in the hardwired layer.

A rather similar approach that I have investigated is the one presented in the paper [77]. The authors aimed to improve the detection performance of the 3D CNN's proposed in [75]. To this end, they explore several architecture variants and compare the obtained results. The idea of this approach is to experiment with various temporal depths of the kernel. These include changing only the kernels of the convolution layers.

The paper explores two types of architectures:

- the 3D CNN-s with homogeneous temporal depth
- and 3D CNN-s with varying temporal depth

In a homogeneous architecture all of the convolution layers have the same temporal depth. To explore the possibilities of such an approach the authors use four network architectures of this type having the temporal depth set to 1, 3, 5 and 7.

The second type includes the architectures that have a kernel with variable temporal depth across the convolution layers. The experiments used in this section are based on two network architectures. The first uses the increasing temporal depth scheme having the depths set to 3-3-5-5-7 and the second uses the decreasing temporal depth scheme having the kernel depths set to 5-5-3-3.

Based on these ideas, the authors tested all the proposed networks on the UCF101 dataset and found that the best architecture is the homogeneous one having the dept set to three. The experiments were caried on the Sports-1M dataset [78]. And the results obtained are decent but not impressive.

The paper [77] also explores the usefulness of the video features extracted by this network. For this purpose, the features provided by the fully connected layer are feed to the multi class linear SVM classifier. This setup successfully captures actions invideo. According to the reported results, the detection accuracy is around 90%, which is very good. However, this solution is hard to train because it requires a huge amount of training data. For instance violent actions in urban areas are very rare, so it is almost impossible to collect such a large amount of training data of this type. The reason it requires a huge amount of training data is that it is mostly based on the appearance of the object. The motion descriptors that are extracted in the hardwired layer are not so descriptive.

The solution to the problem of huge training data would be to use the motion of the object instead of its appearance. This technique is exploited in the approach proposed in [79]. It also uses the 3D CNN network to recognize human actions, but

in a slightly different way. The main advantages of this paper compared to the approach of [75] are:

- the improvement of the detection performance by extending the temporal resolution of the network
- and the improvement of the performance by using quality optical flow algorithms

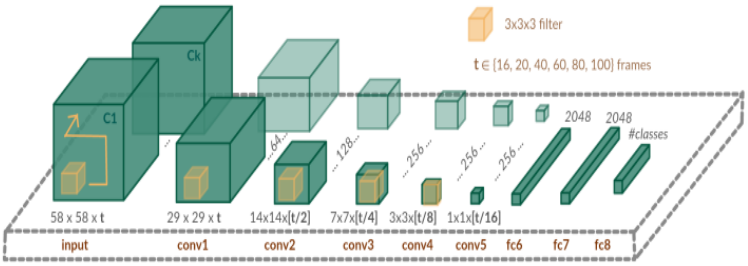


Figure 26 - The convolutional neural network architecture used for recognizing human actions. It consists of five convolutional layers and three fully connected layers [79]

Assuming that the actions usually last a few seconds, thus extending over no more than hundreds of frames, the paper proposes to extend the temporal resolution of CNN 3D to 60 frames. This novel long term temporal convolution (LTC) architecture, shown in the figure 26, improves the performance of the action detection by including more frames into the analysis process. According to the paper, the reason for achieving a lower detection performance of the approaches proposed in [59],[80],[77] is the use of small number of stacked frames in the analysis process. These algorithms process only sixteen frames at a time.

The architecture benefit from the success obtained by the architectures proposed in [77], hence it uses a shape of the filter kernel of 3x3x3. It also defines a novel CNN that consist of five convolution layers, three fully connected layers and one fully connected layer.

In addition to studying the impact of using larger temporal extents on detection performance, the authors investigate the performance of different types of input data. They use the raw RGB, the MPEG flow [81], the Farneback flow and the Brox flow [82]. According to the provided results, the motion descriptors are more effective than the raw image data. The detection accuracy for a RGB input is 59.9 % while for the Brox flow is 79.6 % on the same video. For the Farneback flow and MPEG flow the paper reports lower detection accuracy.

Since the Brox flow best captures the motion, the authors recommend using it with the LTC architecture. This combination achieves the best results, but requires many calculations due to the optical flow extraction algorithm. A smart surveillance camera with limited resources cannot extract this motion descriptor in real time.

Algorithm	Advantages	Disadvantages
Toward abnormal trajectory and event detection in video surveillance	- uses efficent descriptor that is the object trajectories - it uses a gausan modeling techniqu for the classification which is simple and precise	- it uses computationaly expensive descriptors the optical flow and the histogram of oriented gradients
Vision-based human action classification using adaptive boosting algorithm	- uses human siluethes as action features which are easy to extract - uses the AdaBoost classification algorithm which is fast and accurate	- uses a custom background subtraction that do not have a background update procedure so the algorithm will work only in laboratory conditions
The recognition of human movement using temporal templates	- uses two easy to extract motion descriptors that are based on frame difference technique - the classification is simple, it is a template matching technique	- algorithm is very sensitive to oclusions and camera angle
Actions as space-time shapes	- it uses volumetric space time shape of human action that is based on simple human silouethe extraction - it is fast and not computationaly expensive	- it is very sensitive to oclusion and camera nagle
Activity Recognition and Localization on a Truck Parking Lot	- divides the recognition task into subtasks - it detects the activiti zones, the human actions and the formation of group - combines these three features to detect human behavior	- it uses the STIP features for the action recognition which are computationaly expensive
Violence Detection in Video Using Computer Vision Techniques	- it uses the bag of visual words approach which is robust method for	- it uses the STIP and the MoSIFT features which are computationaly

	detecting actions in videos	expensive
Real time violence detection in video	- it has a high accuracy in detecting violent behavior because it uses the motion of objects in the detection process - it uses SVM classifer which is robust and fast	- it uses the optical flow descriptor for the motion estimation , the extraction process consumes a lot of computational resources
Dense trajectories and motion boundary descriptors for action recognition	- it captures accuratelly the human actions - it uses the object motion and the object appearance in the detection process - it uses the robust bag of visual words approach for the classification	- it uses the optical flow, the histogram of oriented gradients and the histogram of oriented flow. All these descriptors require high computing power when extracting from the video
Violence Detection based on Spatio-Temporal Feature and Fisher Vector	- it uses a complex motion descriptor and the MPEG flow descriptor which enables the algorithm to detect precisely the violent actions - it uses the robust bag of visual vords approach for violence detection	- the approcah uses the dense trajectories which are computationally expensive
Two-stream convolutional networks for action recognition in videos	- uses the appearence and motion - has two ConvNets, the first processes the motion and the seccond processes the appearance	- requires huge amount of training data - it uses the optical flow which require complex computations
Beyond short snippets: Deep networks for video classification	- it has multiple CNN, one for frame - combines the appearance and motion in the detection process	- it is hard to train because of using multiple CNN's - it uses the optical flow which requires complex computations
Real-time action recognition with deeply transferred motion vector cnns	- it uses separate CNN's for motion and appearance - it uses a motion descriptor that is not computationally demanding	- requires large amount of training data because of using the appearance in the detection process

Action recognition with trajectory-pooled deep-convolutional descriptors	- defines a efficent video descriptor that combines the optical flow and the feature maps of the ConvNets	- it is computationaly intensive because it requires two CNN's to run in paralel and also uses the optical flow
3D convolutional neural networks for human action recognition	- has a hardwired layer that extracts the motion of objects	- it scores low on detection accuracy
Learning spatiotemporal features with 3d convolutional networks	- it uses quality optical flow for extracting the motion - uses a extended temporal resolution of sixty frames	- it requires high computations due to the extraction of the optical flow

Table 1 - Comparison of video analysis algorithms investigated during the research period.

5 Basic behavior classification in low computational environments. Traffic surveillance application

5.1 Introduction

My first attempt in designing a surveilance system that is aimed for detecting unushual behavior in the crovded cities is aimed for monitoring the traffic surveillance. This system has a capability to automatically detect traffic events and to send them to the control center in order to be further processed by the surveillance officers. In this way, the proposed system can certainly help prevent unusual events from happening, thus providing real-time alerts so that police officers can reach the scene of the event very quickly. Therefore, the proposed system can improve the public sequrity in big cities.This system was a first step towards designing a more complex video surveillance system, which can automatically detect the unusual behavior of the people.In the rest of this chapter I will briefly present the problems encountered in today's traffic and also provide you with information about similar solutions that aim to help solve traffic problems by using video analysis algorithms.

Introduction

We all know that managing traffic in the today's cities is a real challenge. Competitions among car manufacturers influenced the car market. Also the advances in technology made the cars last much more longer compared to the old cars manufactured in the 90 is era. This facts influenced negatively the car prices. Because they are becoming cheaper more people can afford them. As consequence there is a constant car number increase. This fact is inconvenient for large urban areas. Increased number of cars and inadequate infrastructureunavoidably leads to traffic congestion. Hence, this situation introduces nervousnes among the drivers and unavoidabely increases the risk of vehicle accidents. Therefore, drivers caught in a jam are usually becoming nervous.Even some of them may exibit anger or agression behind the wheell.This state of the mind fires the driver to behave inappropriately in a traffic. So, in these situations, these individuals are prone to cause an accident.

Another downside generated by this situation is the inefficient transportation that is the root cause of many problems. For exampledelays imposed by traffic jams have an impact on the goods delivery. If certain goods do not arrive in time they maynegatively affect the industry thus causing additional unecessary costs. This is especially true for medium sized companies that require raw materials for production.

Those are some of thefacts that impose new standards in traffic management systems.Therefore, to meet these novel requirements, city management usually decides to use technology. Mainly the video sensors (surveillance cameras) are used for this purpose. The reason behind this approach is because the cameras are cheap and relatively easy to install. So,this has made the number of video cameras in cities increase day by day thus the surveillance network in urban areas has a growing trend.

This mode of surveillance offers many advantages. For example the video feeds could enable the traffic management system to automatically inform the car driver (ex. via radio, or Android application) about current traffic status. This way the drivers could chose the less congested route. So the unpleasant waiting in a traffic jam could be avoided.

Another benefit from a modern traffic management system relaying on a video surveillance could be adaptive traffic light timing. By dynamically establishing the time of traffic lights the overall flow can be greatly improved. Minimizing the red light time if congestion is detected will greatly improve traffic efficiency. The proposed work in [83] describes the use of surveillance cameras network for adaptive traffic light control. For the ease of installation the authors propose a wireless network. Each camera (network node) is able to communicate with central unit entitled intersection control agent (ICA). The ICA is directly connected to a set

of traffic lights of a particular area. It is engaged for controlling the traffic light timing. It also process the data provided by the network nodes.

The nodes are capable of individually processing the video feeds. Speed, total vehicle number for a particular lane is computed by each node. Embedded video analysis reducers the network bandwidth. Only useful information is transferred to the ICA. For further improvement of the proposed system the authors mention a more advanced system with duplex communication. Moreover, to further reduce the network bandwidth and increase the system efficiency, nodes and the ICA could communicate with each other. This kind of concept is quite interesting because it promises a solution that could solve the traffic congestion problem.

A common problem in traffic surveillance is the inability of surveillance cameras to detect and recognize vehicles at night.Therefore, an interesting solution that addresses this problem and would be very useful for a modern traffic management system is presented in [84]. Interestingly, instead of car shape the authors propose the use of taillights and headlight as vehicle features. Although may appear simplistic this „clever" method may be robust enough. The operation of this algorithm is quite simple.The preprocessing phase applies a multilevel histogram thresholding. Features obtained this way are clustered by a novel method tailored to fit this particular application. Therefore, the algorithm main focus is on clustering the bright objects obtained after the preprocessing stage. Such clustered objects are then classified into either cars or motor bikes. Moreover, in order to improve accuracy the authors use a tracker. The tracker is particullary usefull in cases where there are many oclussions. So by implying it in the detection process the detection errors generated by the oclusions are succesfully solved.

Both forementioned solutions can be successfully used in a modern traffic management system. The [83]is mainly focused on the system. It proposes a novel approach regarding video data collection and data processing. Complementary the work [84] elaborates the object classification in night conditions by using conventional surveillance cameras. It is a good fit for extending the performance of the proposed adaptive traffic control [83]infrastructure.

The data processed by the video analytic could be further used for better traffic management. The routes could be better planned if we know the usual traffic status. A collection of data regarding traffic from certain area could be stored on a central server and made available to the public. This way the users could benefit from it when planning a future route. By future route I mean the route that is planned in advance of several hours to several days. The user can know, with certain precision, all the traffic status like congestion, traffic flow for the trip day. He can chose the most favorable route based on these assumptions. Moreover the collected data could be used for identifying the traffic management issues, generating the statistics and even used for training a deep neural network.

In this regard, the method exposed in [85]presents a solution for classifying and tracking vehicles. The output provided by this algorithm can be easily collected and used for generating long term traffic data statistics. The system is focused on the video analysis rather than on a surveillance system infrastructure. It uses just a single camera for collecting video data. The algorithm proposes a novel solution for recognizing the lanes. When installed the system automatically recognizes the lanes so there is no need for manual selection of the lane delimiters.

The core of the algorithm is based on background subtraction, temporal frame difference and vehicle histogram calculation. The classification stage implies the novel multiple evidence clustering algorithm,whilst, for tracking detected vehicles, the authors use a Kalman filter.Once obtained the tracking trajectory is stored in the local database for futher analysis. Interestingly, beside this common features, the proposed solution is able to classify the vehicles according to their size. It also embeds solutions for maximizing the algorithm performance like algorithms for avoiding shadowing and vehicle occlusions.

All the information acquired by the modern traffic management system can be used for improving the traffic safety. The concept is already taking shape in today modern vehicles. They have numerous driver assistance systems on board that can prevent the driver from making the wrong decisions. Nevertheles, some manufacturers have made it possible to communicate between the vehicle and the control center. On the other hand there is a great effort in enabling the inter vehicle communication. This types of concepts will offer the possibility of providing the driver with information just in time. Thus making the ride much more safer.

One important onboard system regarding traffic safety is the overtaking system. Its purpose is to assist and give the driver instruction about the right timing for making safe overtaking maneuver. The system should be able to combine information regarding the driver skills with the contextual data. Driver skills should reflect the ability of a person to make certain maneuvers. Although this information could be manually introduced (for example beginner/mid level/professional) it is more convenient to be identified by the system. Even if the driver is experienced it may still find difficult do execute specific maneuvering actions. Thus setting manually the driver skill level is not a good choice. If the system identifies correctly the driver level it will certainly give more accurate instruction to the driver when in overtaking.

The overtaking assistance system should collect its data from numerous sources.A high amount of information makes a more accurate prediction.So, combining a large amount of quality information offers the system possibility to correctly calculate the right timing. This data aquisition can be easely done in modern vehicles. Modular organizational structure of the car systems permit easy access to almost all sensors. Hence , in a modern vehicle there are a lot of modules that can communicate to

each other via a data bus and each one is responsible of collecting the signals from numerous sensors connected to it, the sensor data acquired from one module can be easily accessed by other modules by the means of data bus. Moreover the module responsible for collecting data from certain sensors can process data and make it available to the others.This means that, in addition to the raw sensor data, the modules connected to the bus can also access the processed data.This structure allows the novel driver assistance systems to be easely integrated into the car.

Inter-vehicle communication will also contribute to the driver safety. Exchanging data between vehicles will enable the onboard systems to make fast estimation of the traffic situation. If there is a potential risk the system can take just in time actions for avoiding collisions [86]. It can send warning information to the driver or take control over the vehicle. In other words the system can just display a warning message or directly engage the brakes without driver intervention. Another benefit regarding safety is the ability of the driver to see the blind spots. This is particularly useful in the overtaking. For example, the video camera installed on the vehicle to be overtaken could broadcast live streams to the driver who wants to overtake. So the driver who wants to overtake could safely execute the overtaking maneuver.The network used for this type of communication in literature is known as a vehicular ad-hoc network (VANET). Beside inter vehicle communication the framework provides communication with base stations installed on the ground. Moreover, it also offers support for vehicles that are far away from each other. Hence, simple vehicle to vehicle communication is designed for short distances, thevehicles found far away from each other can not exchange data by direct connection. To solve this problem, VANET firstly transfers the data to a base station and then forwardrs itto the vehicle that requested it.

Nevertheless, the technological solutions mentioned in this introductory chapter aim to help modern traffic management systems to cope with the large volume of data collected by the surveillance network. The scope of such systems is to increase the traffic throuthput by reducing the traffic congestion and to increase the traffic safety by reducing the number of accidents. In this regard, I set out to contribute to this field, so I designed a computer vision system for the automatic analysis of video frames from the video surveillance of a crowded intersection.

5.2 The architecture of the proposed system for traffic surveillance

The purpose of the video analysis system proposed in this chapter is to extend the capabilities of traffic management centers. Thus, the surveillance infrastructure and the video analysis algorithm poposed in this section are aimed for increasing the traffic throutput and also can be used for collecting traffic statistical data. Moreover, the solution uses computer vision algorithms that do not require high computational

resources so it can be implemented using smart surveillance cameras that do not have high computantional power. This feature allows installation of several such low-complexity computer vision algorithms on each smart surveillance camera, each with the aim of performing a particular task.For example, the first algorithm is designed to measure traffic flow and the second is meant to detect unusual events, such as car accidents or unalowed u turns at a crowded intersection.In this way, the proposed system can solve several tasks simultaneously without needinghigh endsmart surveillance cameras.Therefore, in the continuation of this chapter I will describe the system architecture and later on I will propose a low complexity video analysis algorithm that is aimed to measure traffic thruoutput and also has the ability to classify detected vehicles in a busy intersection.

That being said, it is obviously that the proposed traffic surveillance system is based on the modular concept. Therefore, the network consist of many smart surveillance cameras, (reffered as network nodes in this chapter) that have a capability to process the captured video on board, the communication tower for establishing the wireless comunication and the control center. The control center contains powerfullservers whose responsability is to orchestrate the comunication within the system. It also contains many monitors that are the tools of the surveillance officers. They use thesemonitors to visualise important events. Hence, for a better understanding, the figure 27shows the block scheme which ilustrates the architecture of the proposed traffic surveillance system.

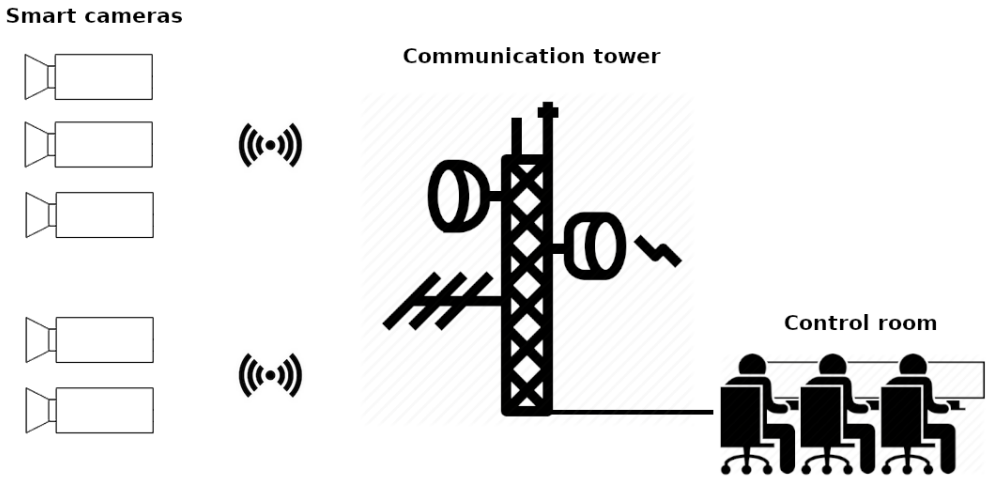


Figure 27 - The proposed video surveillance architecture

The smart cameras proces the video on board so they send only important events to the control center where the surveillance officers analyse and either aprove or disregard the received events.In this way the network throutput is reduced and the amount of video received by the surveillance officers is also reduced. This means

that the surveillance officers analyze only certain events and not the real time video streams that are captured by each of the surveillance cameras. Therefore, even if the smart cameras send a lot of false events to the control room this approach it is still much more convenient than streaming the real time video streams.

The second benefit of the proposed apporach is that it can be easely instaled withouth the need for excesive network wiring. The smart cameras require just a power cable and an antenna to be able to communicate with the comunication tower. Similarly the comunication tower consists of severalantenass that are positioned at the top of it. The tower is ahigh power wireless acces point so in addition to antennas it also contains networking hardware in order to enable the wireless comunication between the smart cameras and the control room. The control room and the comunication tower are conected via cables, but they are very close to each other, therefore these cables are short in lenght and do not require high cost for their instalation.

The smart cameras and the comunication tower should contain networking devices that are able to comunicate over longer distances. Hence, the smart surveillance cameras must have a antena and powerful transimter/receiver in order to be able to receive and transmit data to the communication tower.This involves the use of more power compared to cable communication. When I say cable communication, I am refering to the similar surveillance architecture presented in the figure 27 that instead of wireless comunication uses data cables.

Given the above, using wireless communication and smart cameras, as ilustrated in the figure 27, is a very attractive solution because the system can be easily installed and is cheaper to install than the classical wired version that uses CCTV cameras.Also, it is more efficent because the smart surveillance cameras analyse video on board so only important events are send to the control center for further analysis.This approach improves the chances of detecting events, as it only sends high-value videos to the control center. If the surveillance officers are required to whatch the video streams twenty four seven they will for sure miss many important events. This is because human eyes get tired quickly, especially if they need to inspect multiple video streams at once.So by providing surveillance officers with carefully selected video streams by the video analysis algorithm, the chances of an event being detected are much higher.

These features make such a system attractive formany trafiic surveillance applications. It is especially suitable for the ones that are suposed to detect events, are required to be deployed and set up quickly, and are intended for temporary use. For example, if the traffic management board decides that it would be very useful to measure traffic on an intersection that does not have surveillance cameras installed and there is no AC power available, the proposed system would certainly be the best choice. In such cases, instead of using AC power, smart cameras could use energy

from batteries and solar cells.The communication layer remains unchanged that is the comunication tower and the control center remain as defined in the figure 27. Based on these assumptions, I can say with conviction that this system can be installed easily and quickly even in places where there is no power source available. Moreover, the system can be switched to another intersection at any time, without the need for expensive and complex modifications.

Despite its event detection capability, easy installation and low cost,the proposed system also has its weaknesses.That is, the long distance wireless communication can cause problems due to the fact that the wireless signal may be temporarly disrupted due to interference with wireless signals from other devices.In this situations tha data packests are lost during transmissions so certain events may be lost because of unability to transfer them to the control room.Another downside is that the system does not have the posibility to store videos for longer periods of time. Even though the smart cameras have a local storage and can save videos, the capacity of such local storages is limited. Therefore, surveillance officers cannot inspect old videos, such as those recorded a month ago because they are not found in the camera memory.

Even with the forementioned shortcomings I am trully convinced that the proposed system is still a good choice for the automatic surveillance of a congested intersection in urban areas.If the system is configured correctly, it will definitely meet the expectations of the end user.

5.3 Installation and configuration of smart surveillance cameras

The system I propose in this chapter is an event based one, so that smart surveillance cameras process the video on board and send only important events to the control room. In order to acomplish this task the surveillance cameras need to run an video analysis algorithm. Its task is to analyze video frames in order to be ableto extract useful information from videos, such as traffic irregularities, traffic accidents, etc. This part of the system is very sensitive so it requires a special attention.

In order to get the maximum performance of the proposed system for traffic surveillance the smart surveillance cameras should be installed correctly. It is vital that the camera captures the traffic scene correctly so that the video analysis algorithm can successfully analyze the objects in the scene. In the figure 28is an ilustration that shows how the camera installation should look like. Therfore, my recomandations are that the camera should be installed on the side of the intersection and should be placed on a high pole in order to capture the whole interesection.

This configuration allows the camera to view all important places in an intersection, such as sidewalks and all traffic lanes. Which means that the surveillance camera can be programmed to detect several traffic events at once.When I say programmed I mean that the surveillance camera can run sevaral computer vision algorithms onboard.Each algorithm being designed to capture specific events by analyzing its portion of the monitored scene. This means that the camera can analyze two or more areas simultaneously, so that it can detect for example traffic accidents as well as pedestrians crossing the sidewalks when they are not allowed, that is they have the red light at the traffic light, at the same time.

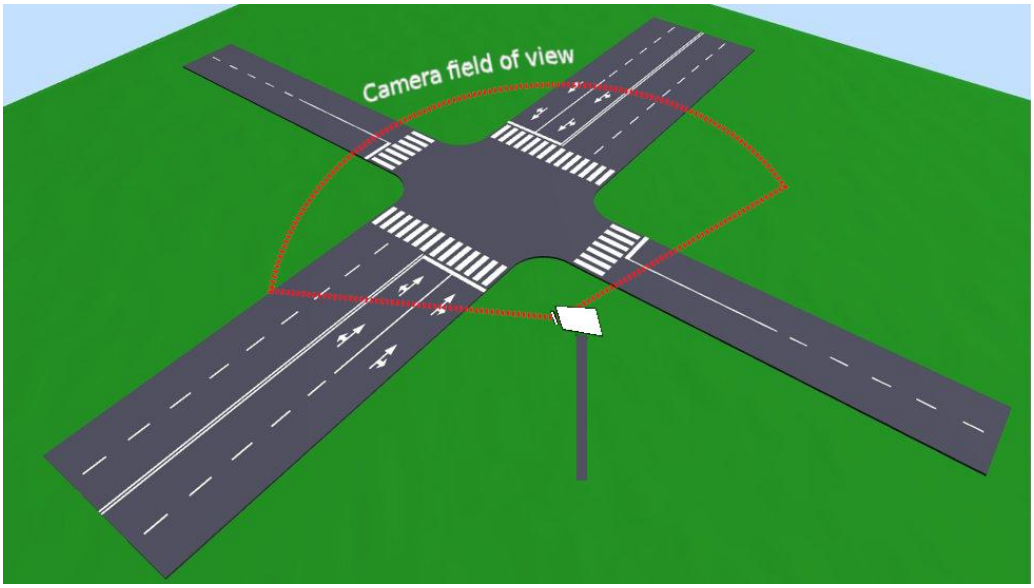


Figure 28 - Position of the smart surveillance camera

The traffic monitoring system I propose aims to automate vehicle detection and classification tasks so it can automatically detect and classify vehicles according to their size. The smart surveillance cameras send only simple messages instead of video frames to the control room. Moreover, the system also has a metering capability, so that, in addition to classification, the algorithm can count the number of light and heavy vehicles passing through the intersection. This kind of information is very usefull for many trafic surveillance applications. Hence, for example it can be used for detecting traffic congestion or to estimate the trafic throuthput.

For achieving this task the proposed system requires just one surveillance camera per intersection. This feature is very convenient because it reduces the cost of the system and also reduces the installation time. The only weak point of the system regarding its installation is the need for a high pole on which the surveillance camera is placed. Hovever, this is not a disantvage because instead of puting the

camera on a pole it can be instead mounted on the clossest building. Of course, this mounting solution is acceptable only if the building is tall and is very close to the surveilled intersection. Otherwise, the system requires the installation of a high pole near the intersection.

In order to function properly, the algorithm also requires a configuration file. That is in addition to the physical installation of the camera, the instalation procedure also requires from the installation technicians to manually select an observation zone within the surveiled scene and save it in the configuration file. An observation zone is the portion of the scene that is used by the computer vision algorithm so it can detect and count vehicles in the intersection. Hence, the purpose of the zone is to „tell" the algorithm where the traffic lanes are, so that the algorithm can correctly count the vehicles in the surveilled intersection. It may have been even more practical for the system to recognize this zone automatically, but this would rise the complexity of the computer vision algorithm. Which in turn means that it would require more computational resources. Hence, I wanted to design an solution that uses as low resources as posible this was not acceptable approach for me.Maybe I could have designed an independent video analysis algorithm to detect the observation zone whichwould have been used only once, that is when installing the camera.

However, during this research work I did not manage to design such an auxiliary algorithm, so the observation zone must be defined manually for each surveillance camera in the system.I consider that this is not a major disadvantage of the system due to the fact that the observation zone needs to be defined only once, that is when installing the surveillance camera.Nevertheless, in the figure 29is an illustration that shows how an observation zone should look like.

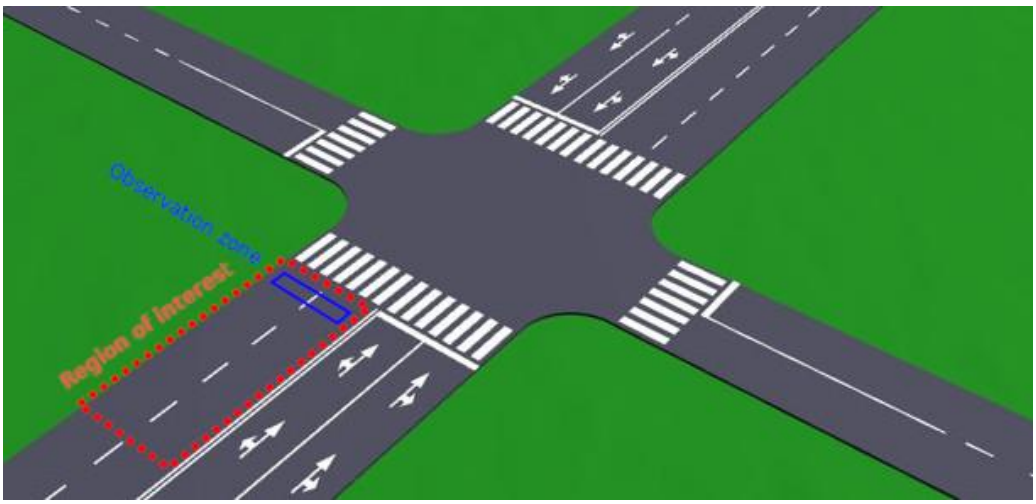


Figure 29 - Illustration of the observation zone and the region of interest

It should be defined in such a way as to include the traffic lanes that are to be monitored by the system as shown by the blue rectangle in the figure 29. The observation zone is used by the algorithm for counting the vehicles. More precisely, the vehicle is counted at the moment it enters the obervation zone.Hence, it is very important to define this zone correctly when installing the surveillance camera so that the algorithm to count vehicles correctly.

The size of the observation zone is not fixed and should have a rectangular shape. Also, the size of such rectangle should be very small in order to enable the algorithm to count vehicles precisely.As a clue the the width of an observation zone should be four times smaller than its height. But, these are just indicative data. In order to get the best detection performance of the algorithm my recommendation is that this area should be established experimentally.
Nevertheless, beside the observation zone the proposed algorithm also requires form the instalation technicians to define and save in the configuration file the region of interest. Indeed, the algorithm works just fine withouth using this region but in that case it analyses the entire video frame so it will inavoidabely consume a lot of computational resources. To avoid such inefficent approach I decided to use a region of interest as illustrated in the figure 29 by the red rectangle. The region of interest is an area defined within the video frame that is meant to filter out the scene zones that are not important for traffic analisys.

The explanation for this approach is that, in order to classify and count vehicles, it is sufficient to analyze only a small part of the video frame and not necessarily the whole frame. That is, it is enough to look at the area that captures only the traffic lanes. The remaining scene zones do not provide useful information to the algorithm so they are of no use for the traffic video analysisapproach proposed in this chapter.

Therefore, for the algorithm to work properly the region of interest must include both trafic lanes of the road segment that is being analysed by the algorithm. To select such a zone I used a rectangle as illustrated in the figure 29. Its size must be considerably larger than the size of rectangle used for the observation zone, as its purpose is to select the part of the monitored scene that captures the route of the vehicles.Based on the knowledge gained during the experiments I can tell for sure that this regionis not as sensitive as the observation area, so even if it has not been perfectly defined, the algorithm will still reach its maximum performance.

Of course, the task of defining the region of interest could also be automated. Similar to defining the observation zone, I could extend the proposed algorithm to automatically recognize the region of interest or maybe even a better solution would be to design a separate algorithm specially designed to perform this task.So the region of interest would not need to be defined manually by the installation

technicians. Implementing such a computer vision algorithm requires a lot of research and is not necessarily a task that should be automated.

The reasoning behind this conclusion is based on the fact that the region of interest requires to be defined only once, that is when the surveillance camera is installed. Therefore the automatic detection of the region of interest will only help technicians during the configuration of the intelligent surveillance camera.It will not in any way extend the traffic analysis capabilyties of the system. Based on these facts I think that it would certainly be nice, but not necessarily useful to have such an auxiliary video analysis algorithm that is meant to automatically detect the region of interest of a busy intersection.

The observation zone and the region of interest explained to this point and ilustrated in the figure 29is the configuration data that is used by the algorithm in the process of detecting, clasifying and counting vehicles that are passing thru just one road segment of the surveilled intersection. This surveillance configuration can be easely extended to monitor more than one road segment of an intersection. In this case the smart surveillance cameras need to run several instances of the proposed video analysis algorithm for traffic surveilance. Each of them beeing responsible for analysing just one road segment. Of course, in order to be able to function propoerly each of the computer vision algorithm should have a separate configuration file that contains the observation zone and the region of interest for the road segment it is analysing. For instance in the case of crossroad ilustrated in the figure 29 if the end user whants to monitor the trafic on all lanes the surveilance camera requires to run four computer vision algorithms each having its own configuration file.

5.4 The proposed video analysis algorithm

In this chapter I will describe in detail my video analysis algorithm that I published in article [BM1] and presented in the first PhD report [BM5].Remember that the target was to design an efficent computer vision algorithm that does not require a high computational power to run so it can be used in a harware with limited resources. This requirement sould not affect its performance so the algorithm should be robust in order to acurattely detect, clasify and count vehicles in an busy intersection.

The algorithm I propose in this chapter satisfies both requirements. This features makes the proposed algorithm be very attractive for using it in the surveillance networks that use smart surveillance cameras for capturing video footages across the city.

The benefits of using smart surveillance cameras in a surveillance network is that in addition to simple video recording, these cameras have the ability to store and process data localy before sending it to the control center.We all know that every benefit comes with a cost. Hence, the disadvantage of processing data localy is due to the fact that the intelligent surveillance camera has a limited processing power and therefore cannot run a complex video analysis algorithm in real time.

Due to this shortcoming whichis imposed by the limited hardware of the camera, it is very important that video analysis algorithm running on the smart camera donot consume a lot of computing resources and in the same time be complex enough to be able to analyze complex surveillance scenarios. With this in mind, I designed a complex video analysis algorithm that consumes very little computational resources. To be able to design such a chalenging algorithm I used several carefully selected video analysis algorithms that work together in order to acomplish the task of analysing the video footages of a busy intersection.

The operation of the algorithm is divided into phases, so each phase fulfills a particular task.This modular design allowed me to design the algorithm easier, as I was able to measure the effectiveness of each new phase I added.Therefore, after a lot of experiments I ended up using the following phases:

- The fitering phase
- The Foregroud extraction phase
- Correction of the foreground mask
- Detection of the vehicle features
- Post processing phase
- Classification and counting phase

All these phases play an important role in the video analysis process. It is essential that you understand how each of them works, so that you can understand how the video traffic analysis algorithm I designed works.In the folowing chapter I am going to describe each algorithm phase in detail.

5.4.1 Filtering phase

The first phase of the algorithm is cocerned with removing the unnecesary information from the video stream. This phase is engadged to filter out the objects that are not relevant for my traffic surveillance application. I am not interested to detect pedestrians, cyclists, buildings and other objects that do not belong to the small or big class.The small class includes small vehicles such as cars and the big class includes large vehicles such as trucks and buses.

Therefore, to avoid the analysis of objects that do not fall into either the small class or the big class, I decided to only monitor the zone that include the traffic lanes. In this way I manage to separate the vehicles from the other objects in the scene. This zone is the region of interest that needs to be defined by the instalation technicians in the configuration file at the moment of installing the smart surveillance camera.

To better understand the principle of operation of the filtering phase, I will explain the process on a concrete case. This use case is the one I am going to use to describe the operation of the algorithm. It is used in the following chapters as well as in the chapter dedicated to experiments for approving the efficiency of the proposed traffic surveillance algorithm. Hence, for testing the proposed algorithm I made a dataset by recording a busy intersection with a video camera. Please note that the details regarding this novel dataset are not important at this moment and are described in one of the folowing chapters.

Nevertheless, the original frame of the recorded intersection is illustrated in the figure 30.aas you can see it contains many regions that are not relevant to this traffic surveillance application. Such as crosswalks, paths for pedestrians and cyclists, buildings, green spaces and parking zones.

a) b)

Figure 30 - a) Original frame as captured by the camcoder [BM1]; b) The extracted region of interest (ROI)

The zone I am interested in is illustrated by the white rectangle in figure 30.a. Looking more closely the zone looks like in figure 30.b. It contains two traffic lanes and two tramway lines. The selected area was manually defined and is referred to as region of interest (ROI) in this manuscript. By including this filtering process I literally cropped the image, thus the algorithm processes just the ROI.

5.4.2 Foreground extraction

The next phase of the algorithm has a task of detecting the moving objects in the video stream. To acieve this task that is to select the best foreground extraction algorithm, in my experiments, I explored three widely used background subtraction algorithms, namely optical flow, frame difference and Mixture of Gaussian (MOG). Even though all of them behave as expected, just one captured my interest.

As far as performance is concerned, the background subtraction algorithm based on optical flow has proven to be very good at detecting moving objects. It successfully managed to separate the moving objects form the background. Nevertheless, despite of the achieved performance, in some cases the algorithm failed. This happens because the algorithm relies solely on pixel motion, so foreground objects are detected only if they are moving. The algorithm failed to detect a foreground object that did not move even if the object stopped for only a few seconds.

The optical flow based background subtraction showed high detection performance if the scene objects were moving, respectively low detection performance if the scene objects were static. Since the video footage's I used in my experiments were captured from a crowded intersection, so that they contained only moving vehicles, this lack of performance in detecting static objects did not bother me. The main reason why I did not use the optical flow algorithm in my application is because of its high computational requirements.

Another algorithm I investigated for extracting the foreground was the frame difference algorithm. Although very simple, this method can disclose useful information about foreground. It provides the boundaries of the moving object. The disadvantage is that the inner area of the object is classified as background. Beside this the approach has issues due to ghosting. It does not handle the shadows well. Due to these shortcomings, the frame difference algorithm cannot be used alone to extract the foreground. Instead, it is suitable for use as additional feature in the foreground extraction procedure.

The background subtraction algorithm proposed in [87] is a better option. This algorithm involves the use of a statistical modeling method, namely it uses several Gausians to capture the intensities of the background pixel. Hence its name Mixture of Gaussians (MOG). Accordingly, the probability density of any background pixel is modelled by the equation (8).

$$p(x) = \sum_{i=1}^{K} \pi_i \, N(x|\mu_i, \sigma_i) \qquad (8)$$

Where K represents the number of Gaussians that are used to model the pixel values, the π_i it's a weight of the i'th component so it is the probability of x belonging to the i'th Gaussian. The μ_i is the mean and the σ_i is the variance of the probability density function Nthat belonghs to the i'th Gaussian. The probability density function for each of the component mixture is defined by the equation(9).

$$N(x|\mu_i, \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \quad (9)$$

Therefore, the parameters of the model that are learnt during training are the π_i , μ_i , σ_i. Tose parameters are learnt by usign the expectation maximization algorithm. It is a iterative algorithm for estimating the maximum likelihood. In each iteration it computes the probabilities of pixels and updates the model parameters. This process continues until the algorithm converges, so it founds a maximum likelihood estimate. In this way the algorithm builds a model of the image pixels by using several Gaussian distributions.

To better understand this concept, in the figure 31 is an illustration of such a Gausian mixture model that uses three Gaussians. Usually, the background pixels appear more frequently in the scene so they are captured by the Gaussians that have high mean value and low variance. These features correspond to the blue and orange bell shaped curve in the figure 31. On the contrary the foreground pixels appear occasionally in the frame so the foreground is captured by the Gaussians that have low mean and high variance such as the green bell shaped curve in the graph.

At first glance you might think that the number of Gaussian components used for modeling follows the rule, the more the better. But this is not true for the MOG algorithm. According to the research carried out in the paper [88],the number of Gaussians should not be more than seven and not less than three. Indeed the algorithm can work with only one Gaussian, but in this case the algorithm does not reach its maximum performance.

Also in the same article I found out that the number of five distributions is the most suitable for background modeling. The use of six or seven Gaussians increases the complexity of the algorithm and does not greatly improve its performance. Therefore, I decided to use five Gaussians in my experiments to model the pixel values.

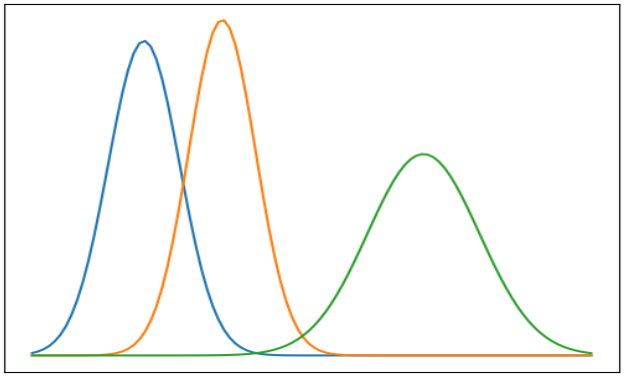


Figure 31 - GMM of a pixel with number of gaussians set to three

An important feature of this algorithm is that it constantly updates its models. So it is able to quickly adapt to the ever-changing background. This feature is very important, especially if the scene takes place outdoors.

Figure 32 - Left original frames , right the foreground masks generated by the MOG algorithm

Another feature of the MOG algorithm is that it can recognize the shadow pixels. It also, successfully handles periodical motions of small areas of the background. Often, the scene contains trees whose leaves are moving due to wind. Even in this case where the background contains objects that exhibit small periodic motion, the MOG algorithm can successfully classify foreground pixels. The algorithm adapts quickly to the scene. It constantly monitors video frames and updates its model in real time. Those features convinced my that the Mixture of Gaussian approach is the best fit for my application.Figure 32 is an illustration of the original frames and their foreground masks, generated by the MOG algorithm.

5.4.3 Correction of the foreground mask

However the MOG has a weakness shared with all currently known background segmentation methods. In many cases, due to the classification errors, some of the foreground pixels are classified as background instead of foreground. Hence, I assume an object, in this application a car, is oftentimes represented by a couple of interconnected foreground regions. Detected foreground often contains holes, where object features are more similar to the background model. In order to overcome the shortcoming of the background subtraction algorithm, that is to correct the foreground mask as much as possible, morphological operations are needed. To restore foreground connectivity and to fill in small holes, I use morphological dilation [89].

Dilation operator is a basic operation frequently used in image processing. It is usually exploited for the gradual extension of the white pixels areas in binary image. The operator is particularly useful for correcting foreground mask imperfections generated by the background subtraction algorithm. It extends the foreground area and at the same time reduces the parasitic holes found in the foreground mask.

The operator receives two inputs, a binary image and a structuring element, also called kernel in the mathematical morphology theory. The dilation combines two sets so its operation is defined by themathematical formula (10) which is known as the Minkowski sum in geometry:

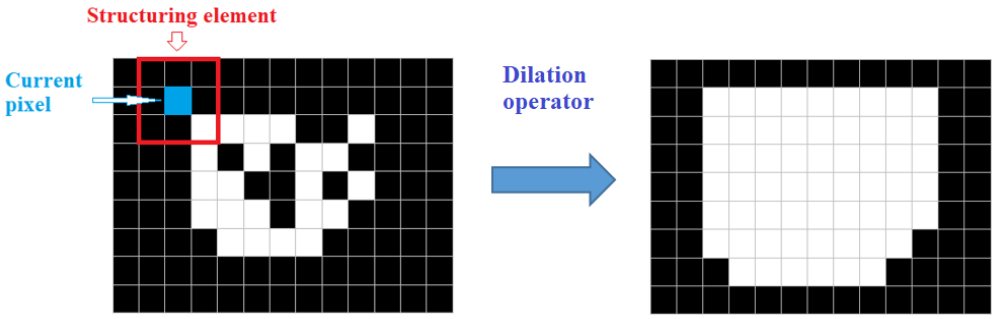
$$A \oplus B = \{a + b \,|a \in A, b \in B\} \tag{10}$$

The A is an image while the B is the structuring element that is it is the structuring element that is used to dilate the image A. Lets suppose the following example. The binary image, illustrated in figure 33.a, is subjected to the dilation operator. To keep it simple, the structuring element is set as having a 3x3 rectangular shape. At each step the algorithm processes just one pixel. The number of iterations is equal to the number of pixels in the image. In each iteration, the center of the structuring element is positioned at the pixel to be processed. The pixel color is changed from black to white only if any pixel of the structuring element matches at least one white pixel in the image.

An iteration of the dilation operator is illustrated in figure 33.a . The pixel to be processed is colored blue and is called the current pixel, whereas the red rectangle shows the structuring element. Initially the current pixel value had a black color. Because there is at least one white pixel within the structuring element, the current pixel changes color to white.

Therefore, the dilation operator makes smaller the black holes found inside a white blob and slightly extends the white blob area. The size of the structuring element

dictates the size of the filling. After applying the dilation operator the image looks like in the figure 33.b.

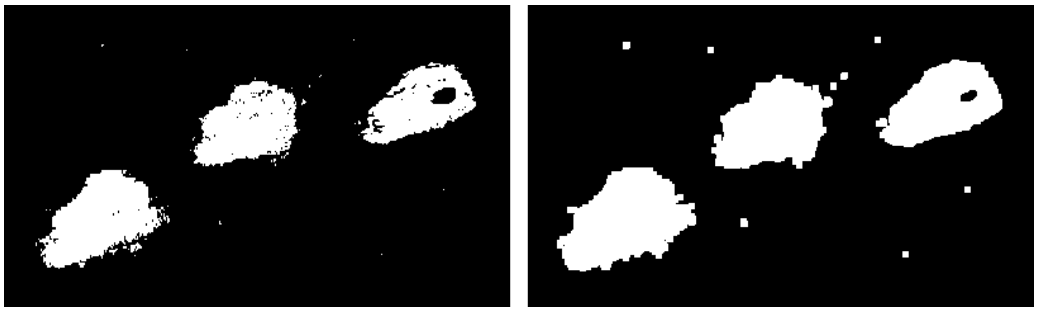


a) b)

Figure 33 - a) The input image; b) The image after applying the dilation operator.

In the proposed traffic surveillance algorithm, the foreground mask is dilated by a 5x5 kernel at each frame. The reason of chosing a rectangular shape for the structuring element is due to the fact that the foreground mask contains many small rectangular black holes whinh the contours that need to be corrected. As the shape of structuring element depends on the objects that are going to be processed I decided to use a rectangular shape for it.

The kernel size defines the maximum gap between segments and the maximum diameter of holes that will be restored by morphological dilation. Because the foreground mask contains a lot of holes that have a diameter larger than three pixels I decided that it is best to use the 5x5 structuring element instead of 3x3. By applying the dilation operation, the segments are more connected as illustrated in figure 34.b. Small areas resulting from occlusions caused by other objects are eliminated. As seen in the example in figure 34.a and figure 34.b morphological dilation tends to increase the area of the detected foreground objects.



a) b)

Figure 34 - a) The foreground mask provided by the background subtraction algorithm; b) Dilated foreground mask

Beside its benefic effects the dilation operator negatively affects the noise pixels. The surveillance camera is not perfect and the captured frames are contaminated with noisy pixels. The noisy pixels are not stable, they are usually randomly distributed over the entire frame and occur in very small regions that are composed of one or several noise pixels. This phenomenon causes the background subtraction algorithm to believe that the noise is actually foreground. Figure 34.a illustrates a typical foreground mask that is contaminated with parasitic pixels. By applying the dilation operator, the noisy regions expand. This is an issue that negatively affects the vehicle detection process and can be relatively easy removed. Hence, in order to correct this issue I decided to use the morphological erosion operator.

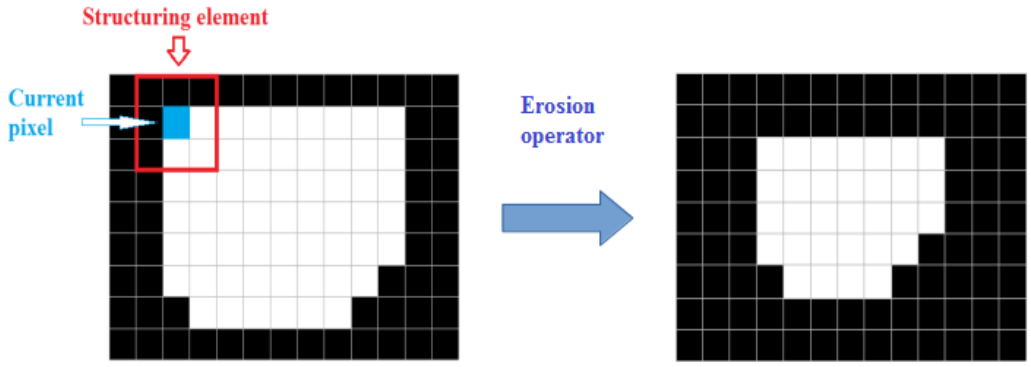
The morphological erosion operator functions in the similar way as the dilation operator. It processes the binary image provided at the input using a structuring element. The difference between the dilation and erosion is that the former is used to increase the image areas, while the latter shrinks the areas. As you may have already noticed, erosion and dilation are complementary morphological operators. The erosion operator is usually applied to binary images, but can also be used to process grayscale images. When applied to the foreground mask, the erosion removes a portion of the mask.

Operation of the erosion operator is defined by the formula(11)which is known as the Minkowski difference in geometry:

$$A \ominus B = \{a - b | \, a \in A, b \in B\} \tag{11}$$

The A is the image and the B is the structuring element that is it is the structuring element that is used to erode the image A. To better understand this concept I am going to describe the process with an example. Hence in figure 35 is an illustration that shows how the erosion operator works. It receives the binary input image namely a) in the figure. The structuring element that is used in the process is set to have a rectangular shape of 3x3. Then, similarly to dilation, the center of the structuring element is positioned at each pixel of the input image. In each step the operator processed only one pixel. Value of the pixel is cleared, namely color is changed from white to black, if one or more pixels of the image that are encompassed by the structuring element have a black color.

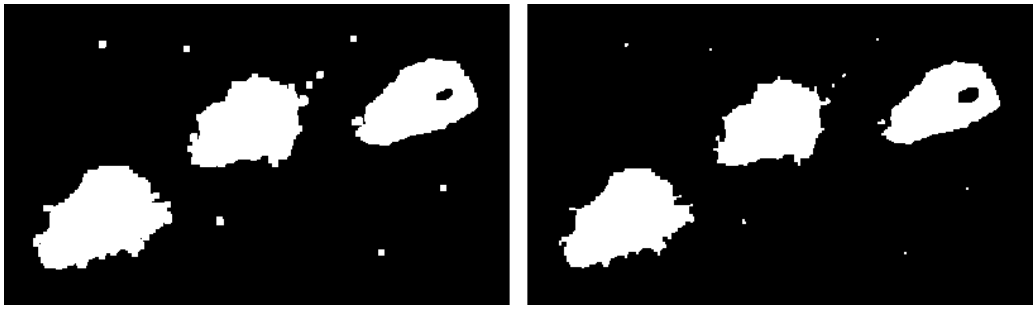
An intuitive example of one iteration is shown in figure 35.a. The center of the structuring element is positioned at the pixel to be processed called the current pixel in the figure. In the initial phase the current pixel had a white color. Since there are black pixels within the structuring element, the color of the current pixel is changed to black. This operation continues until all image pixels have been processed. The result is an eroded image illustrated in figure 35.b.



a) b)

Figure 35 - Erosion operator

Based on the experimental results, I chose to use the morphological erosion operator having the kernel set to a rectangular shape of 3x3. This configuration has proven to be the best for my application. After the operator is applied, all the foreground blobs are shrinked. In figure 36.a is the input image. The small zones that are caused by the dilation of the noise pixels are shrink. Vehicle blobs are also reduced in size. The only disadvantage of the erosion operator is that it tends to enlarge the cavities within the blob. Please note that in figure 36.a, the top right blob has a cavity inside. The same cavity is larger in the eroded image illustrated in figure 36.b.



a) b)

Figure 36 - a) Dilated foreground mask used as input for the erosion operator; b) Mask after erosion

The foreground mask contained noise when it was provided by the background subtraction algorithm. Due to occlusions, some of the pixels in the foreground were misclassified as background. This event caused the apparition of cavities inside the vehicle blobs. Another error of the background subtraction algorithm is due to the camera noise. The noisy pixels were oftentimes classified as foreground.

To tackle these issues I used the morphology operators. The dilation operator filled up the cavities and enlarged the noisy blobs, whilst the erosion shrink-ed the area of all blobs. In this way I managed to remove the blob cavities and to reduce the number noisy pixels.

5.4.4 Detection of the vehicle features

To this point, some of the anomalies generated by the background subtraction algorithm are removed from the foreground mask. The mask is ready for the next processing stage, that is finding the locations of vehicles. In order to fulfill this task it was necessary to find out the center of all the contours of the vehicles encountered in the foreground mask.

The algorithm proposed by Suzuki and Abe described in [90] proved to be very useful in extracting the contours from the foreground mask. It uses a simplistic border following technique that can accurately determine the contour of an object. In the early research phase, I also explored the algorithms for contour extraction described in [91] and [92]. Both solutions use the border flowing technique. But for convenience, I chose to use the Suzuki and Abe's algorithm.

By applying it to the foreground mask I managed to extract the contours of all the objects present in the scene. Figure 37.b illustrates the results. The algorithm received the input binary mask illustrated in figure 37 a.

a) b)

Figure 37 - a) Foreground mask after correction; b) The contours of the vehicles detected by the algorithm of Suzuki and Abe

For the classification of vehicles I chose to use the minimum area bounding rectangle (MBR) approach. It is a fast and robust method for capturing the size of vehicles. The minimum area bounding rectangle of a blob is defined as a rectangle with a minimum area that can fully encompass the blob. An example is illustrated in figure 38. The blobs are colored white and their corresponding minimum area

bounding rectangle is colored green. Also, the figure contains the centers of the blobs. These are illustrated by the red dots.

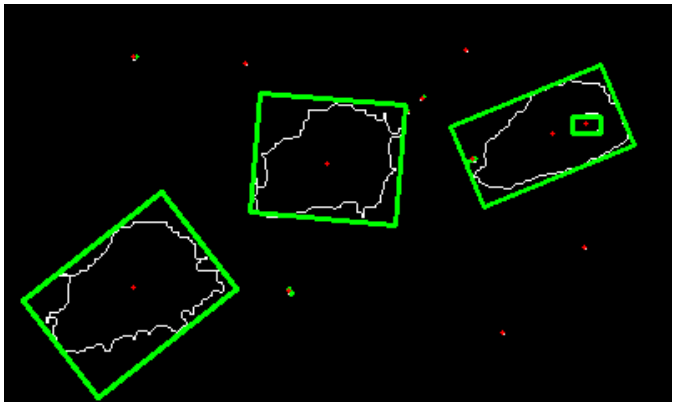


Figure 38 - Minimum area bound rectangles of the vehicles. The red dots in the image are the centers of the contours. The small green contours that do not belong to vehicles are due to the camera noise.

The method is able to accurately estimate the object size by using only its contour. It operates in two steps. In the first phase the method computes the Convex Hull (CH) of the contour. The CH in this case is defined as a convex region that can fully encompass the set of planar points P. The set of planar points is provided by the contour extraction algorithm. It is actually the contour of an object because each contour is saved as a vector of planar points.

Thus, in the first phase, of the computation of the minimum area bounding rectangle, the method approximates the contour of the object to a CH. The approximate CH of an object is much easier to manipulate than its contour. An illustration of the CH of a vector of planar points is given in the figure 39.

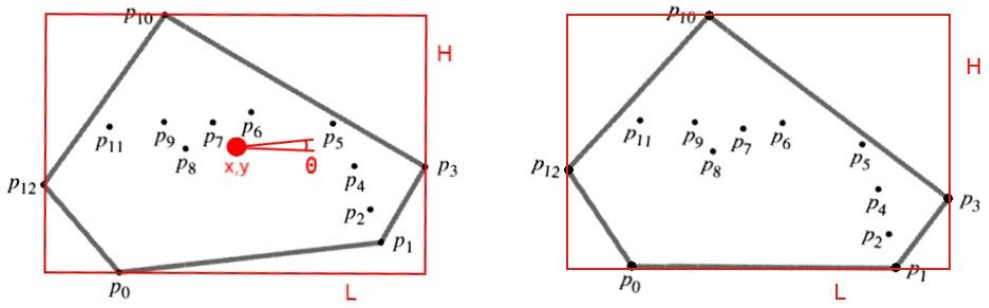


Figure 39 - The process of approximating the vector of planar points to a Convex Hull; a) Computation of the rotation angle θ; b) The Convex Hull and the bounding rectangle in first iteration of the algorithm.

The proposed video analysis algorithm

There are many popular solutions in computational geometry for computing the CH of a set of planar points, like Kirkpatrick–Seidel [93] or the Chan [94] algorithm. Some have linear execution time at best, but their average time complexity is O(nlogh). The *n* is the number of planar points and *h* represents the number of points of the approximated convex hull. The complexity is not stable. It varies between contours because it depends on the number of input points. In the worst case, the parameter h tends to converge to n, so that the time complexity of an algorithm in such scenario becomes O(nlogn).

Since there are many approaches in the literature for calculating the CH, I made the choice only based on the experimental results. Therefore, I decided to use the algorithm described in article [95]. The time complexity of this algorithm is also O(nlogn).

After approximating the contours to CH , the method is ready to calculate the minimum bounding rectangle. This is the second phase of the algorithm. The processing time depends on the number of edges of the convex hull. Thus, the time complexity for this phase is O(n). In order to find the MBR, the algorithm involves several steps. For each of the CH edges, the algorithm computes an orientation θ by the means of the arctangent function. In each iteration the algorithm rotates the CH by angle θ and finds the minimum and the maximum values of the x and y.

These values represent the candidate bounding rectangle of a CH. The CH and the candidate bounding rectangle in the first iteration show as in figure 39.b. Once found, the algorithm stores each rectangle in a buffer. In the last step, the candidate bounding rectangles are compared with each other. The algorithm selects the rectangle with the smallest area as MBR of the contour.

Afterwards the completion of the second phase, each of the contours in the image is assigned a minimum area bounding rectangle. That is, instead of the vector of planar points each contour is described by a centroid C(x,y), an orientation θ, and an area A(L,H) of the MBR. *L* is the width and *H* is the height of the MBR.

Although fast and robust, the classifier also has disadvantages. It fails to discriminate the types of cars. This could be a significant problem for other applications. In the proposed work, I wanted to classify the detected vehicles in only two groups. The first group consists of light vehicles such as cars, SUV's and motorcycles, while the second goup comprises big vehicles like, tramway, trucks and buses. Therefore, the inability of the classifier to discriminate the types of cars did not bother me. Since it is fast and robust and can accurately recognize small and big vehicles, I decided to use it in my application.

5.4.5 Post processing phase

In addition to ignoring areas that are not of interest to this application, it was also necessary to filter out the detected contours. At some point the surveilled scene contained objects other than those present in the small and big class. Pedestrians, dogs and cyclists often appeared in the selected ROI. As a result, the collection contained contours that did not belong to the vehicles.

In addition to this problem, background errors also affected the contour collection. Oftentimes the collection contained small contours that did not belong to any object as illustrated in the figure 38. Those are the contours of the foreground areas that were caused by the camera noise.Therefore, in order to solve these problems, I decided to filtered the contours. I removed items from the contour collection that have a bounding rectangle area smaller than a threshold. The trickiest part in this process was to establish the value of the threshold. But, after a few experiments I was able to easely spot the corect value for this parameter, that is five hundred. Please notice that this parameter depends on the monitored scene. For some scenes, the value of five hundred may not be appropriate. In these cases, the threshold value must be determined experimentally.

Beside the noisy contours the foreground mask suffered from the camouflage issue. This error is also generated by the background subtraction algorithm. Object pixels that are similar in color to the background were often classified as background pixels. This phenomenon caused the formation of cavities inside a contours. Usually, an interior part of the vehicle that was similar in color to the traffic lane was missing from the foreground mask.

To correct this problem, I added another filtering element to the algorithm. The filter removes all contours that have their centroid inside another larger contour. The experimental results have confirmed that this approach is very effective. By applying this process to the contours collection, I managed to eliminate many contours generated due to the camouflage issue.

Segmentation performance was also affected by the camouflage problem. In many cases, the object was captured by two or more contours instead of one contour. One of such cases is illustrated in figure 40 . Usually the contour of an object is detected right and afterwards it splits into two or more contours. A typical solution to this problem would include a tracking algorithm and a more precise vehicle feature descriptor. The idea of this approach is to identify and track the detected objects. However, this concept was not acceptable for my application because it requires the design of an advanced tracking algorithm.

The contour splitting issue generated a lot of miss classifications. Although morphological image processing alleviates the problem, it was necessary to design a correction method.

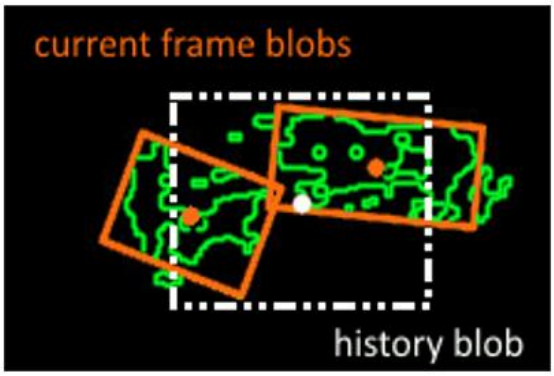


Figure 40 - Illustration of the proposed correction method. The rectangle illustrated by the white dot dashed lines represent the MBR of the history blob. Similarly, the rectagles drawn by the orange lines are the MBR's of the splited blob.

The method is based on the indivisibility and linear motion theory of the vehicle. In reality, the contour of the vehicle is not possible to break. So if the algorithm detected a contour in the previous frame and several smaller contours in the current frame at similar positions, probably the contours belong to the same vehicle. Another feature is that vehicle trajectory is mainly linear. Of course there area also exceptions to this rule. In some situations the vehicle may stop. But these situations are rare, so I didn't take them into account.

Based on this assumptions, I have devised the following simple solution for correcting the contour splitting problem. The method labels two or more contours detected in the current frame as candidates for merging if the following conditions are met. First, the centroids of the MBR's in the current frame must be found within the boundaries of the MBR of larger contour in the previous frame. Second, the area sum of the MBR's in the current frame should converge to the area of the MBR in the previous frame.

Only when these two conditions are satisfied, the candidate contours are merged. This results in a new MBR with the following properties:

- the orientation is the same as the orientation of the MBR in the previous frame
- the size is equal to the sum of the MBR’s areas in the current frame
- the centroid C(x’,y’) is the center of the merged MBR’s in the current frame

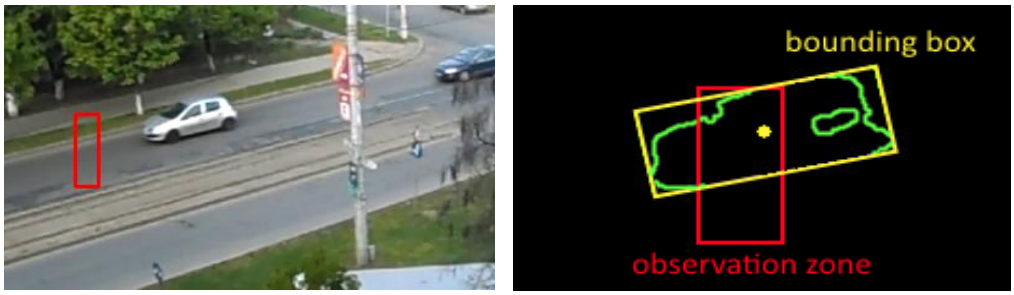
An illustration of such a case is shown in the figure 40. The MBR of merged contours is represented by the white rectangle and the MBR's of contours detected in the current frame are illustrated by the orange rectangles.

Later on, I managed to improve the correction method. Instead of just using the previous frame, I realized that the performance increases if I use several consecutive frames. The improved version of the correction method uses a larger history buffer to store the contour of vehicle and its corresponding MBR.

5.4.6 Classification and counting phase

The next phase of the proposed traffic surveillance algorithm contains a procedure for counting classified vehicles. To accomplish this task I defined a rectangular observation zone in the ROI. This zone is illustrated by the red rectangle in figure 41.a. Once the mass center of the vehicle's bounding rectangle passes through the observation zone, the algorithm increments the vehicle counter by one.

Since some vehicles move slower than others some of them may cause the vehicle counter to be incremented more than once. This situation is present when the center of the same vehicle is within the observation zone for more than one frame. In addition to the vehicle speed, the value with which the vehicle counter is incremented also depends on the camera framerate.



a) b)

Figure 41 - a) The original frame with the observation zone. The zone is illustrated by the red rectangle in the image; b) Detected vehicle passing through the observation zone.

To correct the counting issue I added an inactivity counter to the algorithm. Its purpose is to count the frames that do not contain activity in the observation zone. The inactivity state is present when there is no vehicle centroid inside the observation zone. Otherwise , the activity is detected whenever a vehicle bounding box mass center is within the observation zone. The idea of the proposed method is to wait a few moments after detection in order to allow the vehicle to exit the observation zone. For this purpose I defined a minimum inactivity parameter I_{min}.

Figure 41.b illustrates the moment when a mass center of the vehicle bounding box is in the observation zone. Let's suppose that the mass center just entered the zone and the inactivity counter $I > I_{min}$. In this case the vehicle counter is incremented by one and the inactivity counter is reset. The inactivity counter is constantly reset until the mass center exits the observation zone.

In order to count the next vehicle, there must be inactivity in the observation zone for at least I_{min} frames. Hence, the method is waiting for the vehicle to exit the zone. Afterwards it waits for at least I_m inactivity frames. Only at this point, it is ready to accept the next vehicle. This procedure enabled me to solve the vehicle counting issue. When detected, the vehicle is counted only once.

The I_{min} parameter is very sensitive. It depends on the camera framerate and the speed of the vehicle. Therefore, the optimal value for the I_{min} parameter must be determined experimentally. For this application I found out that the value thirty is the best fit for the I_{min} parameter.

Finally, the last phase of the traffic surveillance algorithm is responsible for the classification of vehicles. I aimed to classify vehicles in only two categories. The light and the heavy ones. To accomplish this task I designed a simple classifier that can separate heavy vehicles from light vehicles. It uses the area of the vehicle bounding rectangle and a threshold. These two values are compared. If the area is greater than the threshold, the vehicle is classified as heavy. Otherwise, the vehicle is classified as light. Mathematically the operation of the classifier is described by the equation (12).

$$L(x) = \begin{matrix} 1, A > A_{thresh} \\ 0, A < A_{thresh} \end{matrix} \quad (12)$$

The A is the area of the vehicle bounding rectangle whilst the A_{thresh} is the threshold. The value 1 is the label for the heavy vehicle, and the value 0 is the label for the light vehicle. Since the camera zoom establishes the vehicles sizes it was necessary to experimentally extract the A_{thres} value.

5.5 The dataset

For testing the traffic surveillance algorithm I used two custom datasets. Both were recorded by me. The first one consists of a video recording of a crowded intersection in Timisoara. It is the intersection between Str. Stefan Cel Mare and Str. Stefan Octavian Iosif. The traffic on these streets is very intense because many heavy and light vehicles pass through them every day. This is exactly what I needed to test the algorithm.

To capture many traffic patterns, I recorded the video at different times of the day such as during rush hour when traffic is congested, during the morning when many heavy vehicles cross the intersection and during the afternoon. At first I captured four video sequences that last approximately ten minutes each. Afterwards I combined all four clips and so I created only one video that lasts forty-one minutes.

The figure 42 ilustrattes several frames from this novel dataset. Each frame is from a separate video sequence.

Figure 42 - Samples from the first dataset

For my experiments I used only the road pointed by the red arrow. In the forty-one minute video on these two lanes of the road, I managed to capture a total of 490 vehicles of which 422 are cars, 4 are bicycles, 34 are vans, 6 are buses and 24 are trucks. In addition to vehicles there are also many other objects in the dataset such as pedestrians and trees. These objects were not of interest to me, but I could not avoid them because they are part of the scene.

I recorded the video from a balcony of an apartment that is located near this intersection. The camera used to record the video was a Samsung WB210 with a video resolution set to 1280x720 pixels and a frame rate set to 30 frames per second.

The dataset

The second dataset I made is more challenging for the algorithm. In addition to the large number of large and small vehicles, it also contains video sequences that are captured in low light conditions. Similar to the previous dataset, I recorded four video seqences at diferent times of the day, each lasting about ten minutes. But this time I wanted to capture the scene under various lighting conditions.

In the figure 43 are samples from the captured video sequences. The left frame in the first row is from the video recorded at 9AM, while the right frame is from the video recorded at 13h. Samples in the bottom row of the figure 43 are from videos that are recorded in the afternoon of the same day. The frame on the left is from the video recorded at 18h, and the one on the right is from the video recorded at 23h.

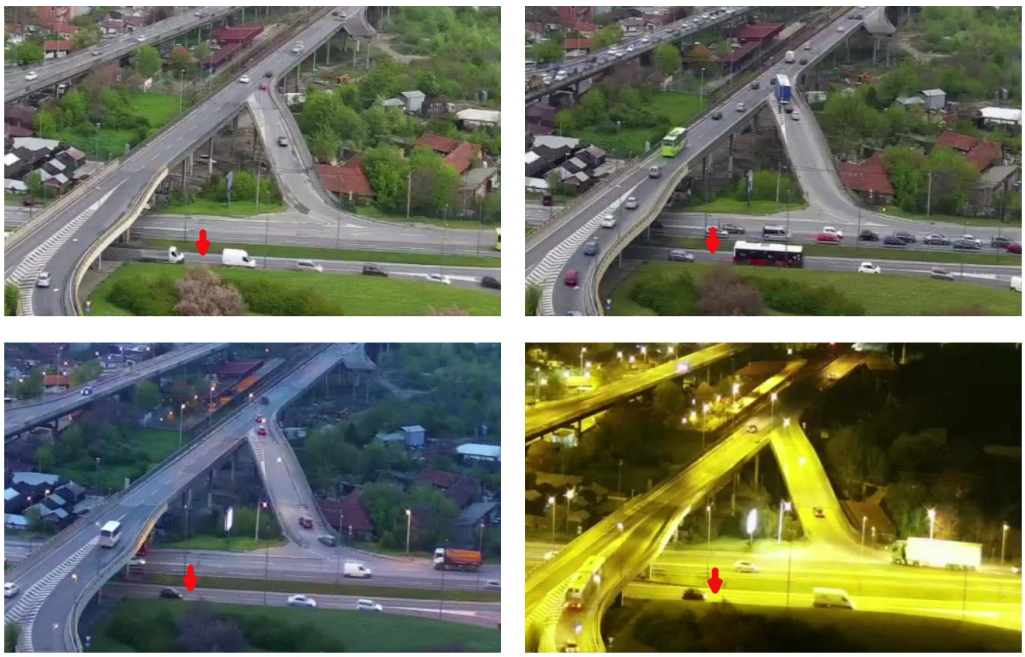


Figure 43 – Samples from the second dataset.

The frames in the figure 43, as well as the four recorded videos, capture traffic from a very crowded intersection in Belgrade. It is the loop of the Pancevo Bridge that connects the Pancevo Bridge with Despot Stefan Boulevard. This intersection contains many roads, but only some of them are suitable for use by the algorithm. This is because the algorithm is designed to analyze vehicles that are captured from a side view.

In order to fulfil the algorithm requirements, I selected only a portion of Despot Stefan Boulevard. It is the road shown by the red arrow in the frames in figure 43. In this road segment which is composed of two traffic lanes I managed to capture:

- for the video recorded at 9AM which lasts 11 minutes and 12 seconds , 164 small vehicles and 25 large vehicles
- for the video recorded at 13h which lasts 9 minutes and 21 seconds, 407 small vehicles and 24 large vehicles
- for the video recorded at 18h which lasts 12 minutes and 9 seconds, 146 small vehicles and 14 large vehicles
- for the video recorded at 23AM which lasts 10 minutes and 33 seconds, 77 small vehicles and 10 large vehicles

All four videos that make up the dataset are captured from the same location, that is, from the roof of a nearby tall building. To record them I used the Samsung WB210 camera with the resolution set to 640x480 pixels.

5.6 Experiments and results

To validate the approach, I used in my experiments the two novel datasets, the openCV computer vision library and the C++ programming language.

The user interface that I used to capture data provided by the algorithm is a graphical interface provided by the openCV library. It is simple and very easy to use. It is based on the imshow function that is aimed to display a video frame.

Figure 44 – User interface of the traffic surveillance algorithm

Prior to displaying the frame the user can add geometric shapes to it as well as text. For my experiments I drawed a small red rectangle in the frame by using the openCV rectangle function with the scope of displaying the observation zone on the user interface. Moreover, in the top left corner I drawed the value of the counter for small vehicles and the value of the counter for large vehicles by using the putText function of the openCV library. I was preaty satisfied with the results obtained, the appearance of the interface is quite decent and practical as you can observe in the figure 44.

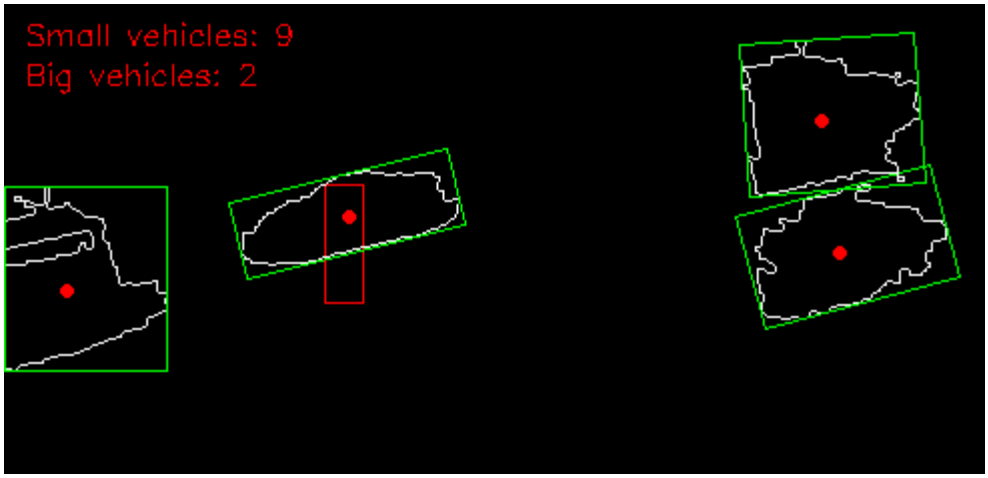


Figure 45 - Graphical interface used for visualizating the vehicle features.

Beside the user interface from the figure 44 I managed to display another interface (figure 45) which is aimed to show the algorithm features in action. With the use of it I managed to fine tune my algorithm because it displays the vehicle contours alongside with the vehicle's bounding boxes. It also contains the observation zone and the value of the contours for small and large vehicles. I used the same principle as for constructing the first interface. That is the imshow function to display the frame, the rectangle function to draw observation zone, the putText to draw the value of the counters, the circle to draw the centroid of the bounding rectangles, the polylines to draw the contours and the line to draw the minimum bounding rectangles.

5.6.1 Experiments performed on the first data set

The frame displayed in the figure 44 is just a region of the original frame. Remember that, the algorithm processes just the zone where the vehicles travel and ignores the remaining portions of the frame. In order to do so it requires from the user to define the parameters of the region in the configuration file. In my experiments, I realized that for the first dataset the region of interest that captures traffic lanes is best defined by a rectangle that has the following coordinates:

- top left point x=160, y=245
- bottom right point x =650, y=480

The resolution of the video frame from which the region of interest is extracted is 1280x720 pixels. The coordinates x=160 and the y=240, of the top left point, represents the horizontal and vertical distance in number of pixels from the top left corner of the video frame. The same rule applies to the bottom right point.

Beside the region of interest the algorithm requires from the user to define the coordinates of the observation zone. This zone is also of rectangular shape. In my experiments it has the following coordinates:

- top left point x=160, y=90
- bottom right point x =180, y=150

Similar to the region of interest, these coordinates are absolute coordinates. They represent the positions of the vertices of the rectangle relative to the origin of the region of interes frame. In openCV, the region of interest is treated as a separate frame with the origin in the top left corner.

In the figure 46 is the original frame and the two rectangular zones. The red rectangle is the region of interest and the blue rectangle is the observation zone.

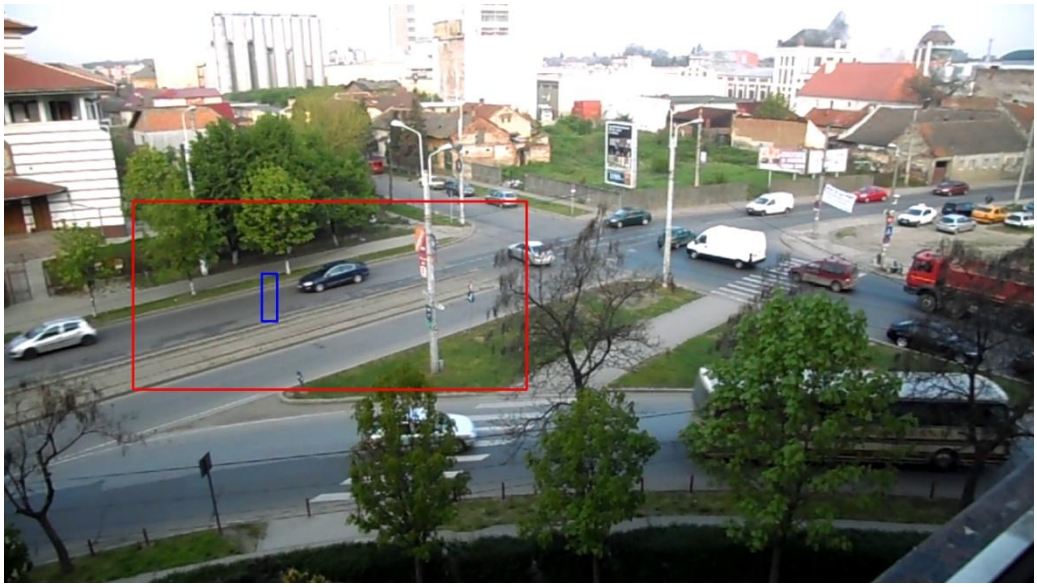


Figure 46 - Original frame of the first data set with the region of interest (red rectangle) and the observation area (blue rectangle)

Experiments and results

In addition to specifying the parameters of the region of interest and the observation zone the algorithm requires the user to define three configuration variables. That is the area threshold variable, the inactivity counter threshoold variable and the correction history buffer size variable. The area threshold is expresed in number of pixels and it is used by the algorithm to make diference between small and large vehicles. If the vehicle size exceds the area threshold variable, the algorithm classifies the vehicle as belonging to the large vehicles class. Otherwise the algorithm classifies the vehicle as a belonging to the small vehicle class. Obviously, the size of small and large vehicles depends on the distance between the surveillance camera and the vehicles. The greater this distance, the smaller the vehicles appear in the video and vice versa. Therefore, this parameter depend on the environment and must be established experimentally. It is not affected by the inactivity counter threshoold and the history buffer size, so this is the first configuration variable to be set in the algorithm configuration process.

In order to find the correct value of the area threshold variable I conducted the folowing experiment. I chose six vaues ranging from 2000 to 7000 with increments of 1000 as specified in the table 2. Then I selected a portion of the video in which a small vehicle (a car) and a large vehicle (a truck) pass through the observation zone. For each of the six values I inspected the counters of the algorithm, so that I wanted to find out if the classification goes well.

	Threshold values(expressed in number of pixels)					
	2000	3000	4000	5000	6000	7000
Small vehicle counter	0	0	1	1	1	2
Large vehicle counter	2	2	1	1	1	0
Observation	The small vehicle is missclassified	The small vehicle is missclassified	All vehicless are classified corretly	All vehicless are classified corretly	All vehicless are classified corretly	The large vehicle is missclassified

Table 2 - Results of classification on the first data set for different values of the area threshold variable.

The values 2000 and 3000 are too small and the algorithm classifies both vehicles as belonging to large vehicle class. If the value of thresold is set too high like when

using the value 7000 the algorithm does the opposite that is it classifies the large vehicles as belonging to the small vehicle class. It seems that the values 4000, 5000 and 6000 are the right ones because in these circumstances the algorithm correctly classifies both vehicles. Just to be safe, I decided to use the 5000 value for the area threshold variable.

The seecond configuration variable that is required to be established experimentaly is the inactivity counter threshold. It is used by the algorithm to avoid counting a vehicle multiple times. As you may remember from chapter 4.4.6, it is a waiting time used by the algorithm which is activated when a vehicle exits the observation zone. Its purpose is to block vehicle counters for a period of time so that the algorithm can count vehicles correctly.

Inactivity counter threshold is expressed in number of frames so its value depends on the framerate of the camera. As such it must be established experimentally. It should be neither too small nor too big. In my experiments, to determine its value, I inspected the results of the vehicle counting reported by the algorithm for different threshold values as shown in the table 3.

	Inactivity counter threshold				
	1	2	3	4	5
Small vehicle counter	600	575	522	382	348
Large vehicle counter	52	48	26	25	19

Table 3 - Results of classification on the first data set for different values of the inactivity counter threshold variable.

For values one and two the algorithm counts the same vehicle multiple times. This is because the centroid of some vehicles are not stable. After the centroid of a vehicle exits the observation zone it may shift back in the inside of the obervation zone for a few frames. As the values one and two are too low for the thresold, the algorithm waits too short after one vehicle leaves the observation zone and thus counts the same vehicle more than once. This is the reason why the number of small and large vehicles are too high for the thresold values one and two.

For the thresold values four and five the number of small and large vehicles are too low. In this situations, the algorithm waits too long after one vehicle exits the obervation zone and misses counting the next vehicle. The results that are close to the ground truth are when the inactivity counter threshold is set to three. This configuration seems to be the best choice, so I decided to select it for my experiments.

Experiments and results

The last configuration variable that needs to be determined experimentally is the size of the correction history buffer. This variable is used by the algorithm correction method which aims to correct the problem of contour splitting of a vehicle. In many cases, instead of one contour a vehicle is represented by two or more contours. This phenomenon has a negative impact on vehicle counting, as some vehicles are counted more than once. To overcome this problem, the corection method detects the contours belonging to the same vehicle and joins them. In this way, the vehicle that was before the correction represented by several contours it is after the correction represented by a single contour.

The corection history buffer plays an important role in the correction procees. Its size determines the number of previous frames that the algorithm will take into account when performing contour correction. According to the experimental results shown in table 4 the size of the correction history buffer does not make sense to be large. Indeed, there is a slight improvement in detection performance from size two to size three. But for larger buffer sizes like for five ten and fifteen the detection performance remains unchanged. This indicates that the background subtraction accoplishes its task sucesfully, so that only in one or two consecutuve frames the vehicle is representred by several smaller contours instead of a large one. Based on these results I conclude that there is no need for using large buffer size. The buffer of size three is sufficent for the correction method to perform its task sucesfully.

	Correction history bufer size				
	2	3	5	10	15
Small vehicle counter	520	516	516	516	516
Large vehicle counter	27	29	29	29	29

Table 4 - Results of classification on the first data set for various sizes of the correction history buffer variable.

With the regions of interest and the configuration variables set, the algorithm achieves a very good performance in detecting and counting vehicles. The graph in figure 47 illustrates the number of small and large vehicles counted by the algorithm and the ground truth that is the actual number of small and large vehicles in the first dataset. The algorithm succeeded to detect almost all vehicles. It failed to detect only the vehicles that stopped. This was to be expected, because the background subtraction algorithm classifies the vehicles as background shortly after they stopped. The problem can be alleviated by working with a lower learning rate for the MOG, at the expense of a slower adaptation for illumination changes. Another possible solution without altering the learning rate is to detect stationary foreground and to suppress background model updating for pixels belonging to stationary background.

The experimental results show that for the first dataset the error ratio of the proposed algorithm for traffic surveillance is as follows:

- +12.17% or +56 for small vehicles and
- -14.70% or -15 for large vehicles

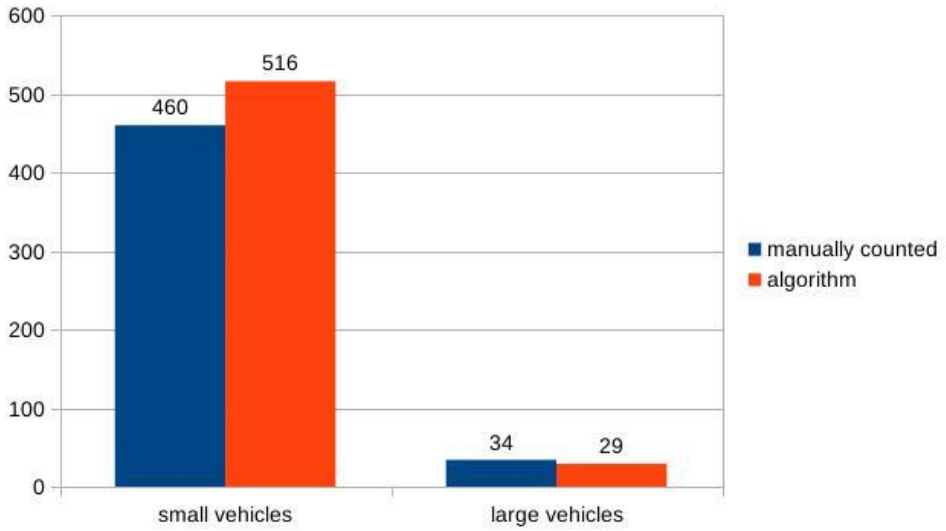


Figure 47 - Classification results and ground truth for the first dataset

These data reveal us that the results obtained are good enough so that the algorithm can be used safely in real traffic surveillance applications. The values of the error ratio which is + 12.17% for small vehicles and -14.70% for large vehicles indicate that the algorithm detected a few extra small vehicles and missed the detection of several large vehicles. Hence, the detection performance is not perfect but given that the algorithm is robust and is not computationally expensive these results are excelent.

5.6.2 Experiments performed on the second data set

The second data set aims to test in more detail the detection performance of the traffic surveillance algorithm. It consists of four video sequences each recorded in different lighting conditions. This scenario is meant to show us how much the lighting conditions of the scene affect the detection performance of the algorithm.

Before testing, I had to go through the configuration process. I established the new values for the region of interest, the observation zone and the configuration

variables. I could not use the configuration parameters from the first dataset, because each scene is unique and requires its own parameters.

The region of interest needs to capture a portion of the road where the vehicles are captured from the side. To meet this requirement I chose for the region of interest the zone delimited by red rectangle in the figure 48. It is a portion of Despot Stefan Boulevard where vehicles are captured from the side. The coordinates that I used to extract this region are:

- top left point x=320, y=380
- bottom right point x= 580, y=430

After defining the region of interest, I proceeded with the definition of observation zone. This zone must be small in width, must cover the entire height of the region of interest and must be within the region of interest. It should also be positioned away from stationary objects, such as light poles, to avoid occlusion of vehicles with these objects. The perfect place for the observation zone is the area delimited by the blue rectangle in the figure 48. It meets all the above requirements. The coordinates of the two points I used in my experiemnts to select the observation zone are:

- top left point x=130, y=5
- bottom right point x=150, y=45

The coordinates of this two points are the distances relative to the upper left point of the region of interest.

Figure 48 - Original frame of the second data set with the region of interest (red rectangle) and the observation area (blue rectangle).

In the next stage of the configuration process I defined the configuration variables using the video sequence recorded in the morning at 9AM. For the area threshold variable I followed the same experiment as for the first dataset. That is, I inspected the output of the algorithm for six values of the variable, but this time ranging from 500 to 3000 with steps of 500 as specified in the table 5.

	Threshold values(expressed in number of pixels)					
	500	1000	1500	2000	2500	3000
Small vehicle counter	0	0	0	1	1	2
Large vehicle counter	2	2	2	1	1	0
Observation	The small vehicle is missclassified	The small vehicle is missclassified	The small vehicle is missclassified	All vehicless are classified corretly	All vehicless are classified corretly	The large vehicle is missclassified

Table 5 - Results of classification on the second data set for different values of the area threshold variable.

The values of the threshold variable chosen to perform the experiment are lower compared to the values used to test the first data set. These smaller values are due to the fact that the distance between the camera and the portion of the road analyzed by the algorithm is greater compared to the same distance of the first data set.

Nevertheless, according to the experimental results, only the values 2000 and 2500 are appropriate for the area threshold variable. Values between 500 and 1500 are too low, while the value 3000 is too high. So, for testing the second data set, I chose to use the value 2000 for the area threshold variable. I consider that the 2500 is too high, because some large vehicles, such as small trucks, do not have an area larger than about 2200 pixels.

To define the second configuration variable that is the inactivity counter threshold, I used the same principle. I chose five values between one and five and inspected the output of the algorithm for each of them.

Experiments and results

	Inactivity counter threshold				
	1	2	3	4	5
Small vehicle counter	203	184	154	142	140
Large vehicle counter	35	18	16	16	15

Table 6 - Results of classification on the second data set for different values of the inactivity counter threshold variable.

In the table 6 are the results obtained. As you can see, the value two provides the results that are closest to the ground truth. For this value the algorithm reports 184 small vehicles and 18 large vehicles. These numbers are very close to the actual number of vehicles in the test video sequence. That is the ten-minute video recorded at 9 AM which contains 164 small vehicles and 25 large vehicles.

For the remaining values, the results are not so good. Value one makes the algorithm count more vehicles than they are in the video. Values three, four and five also negatively affect the counting process, so the algorithm reports fewer vehicles than are in the video. Therfore, the only value that provides the best results is value two. So I used this one in the dataset testing process.

The last stage of the configuration process involves setting the correction history buffer variable. To find the optimal value for this variable I proceeded as in the case of the first two variables. That is, I chose five values and inspected the output of the algorithm for each of them, as shown in table 7.

	Correction history bufer size				
	2	3	5	10	15
Small vehicle counter	180	178	176	176	176
Large vehicle counter	20	21	22	20	20

Table 7 - Results of classification on the second data set for various sizes of the correction history buffer variable.

I noticed that as the buffer size increases, the results get better, but this is only valid for buffer sizes of up to five. For buffer sizes larger than five, the classification performance does not improve any further. Thus, I decided to use the value five for the correction history buffer size variable.

At this point the configuration process is completed, so that the algorithm is ready to be tested on the second dataset. To test it, all I had to do was give it the path to the configuration file and the name of the video. The configuration file contains the region of interest, the observation zone and the configuration varibles.

		Video 1(09h)	Video 2(13h)	Video 3(18h)	Video 4(23h)
Algorithm	Small vehicles	176	421	140	40
	Large vehicles	22	21	12	6
Ground truth	Small vehicles	164	407	146	77
	Large vehicles	25	24	14	10
Error ratio	Small vehicles	+12 +7.31%	+14 +3.44%	-6 -4.11%	-37 -48.05%
	Large vehicles	-3 -12%	-3 -12.5%	-2 -14.29%	-8 -40%

Table 8 – Algorithm testing results on the second dataset

Because the dataset contains four video sequences I run the algorithm four times. Each time with a different video file. After each run I gathered the results and thus I built table 8. The classification results are show in the row labeled Algorithm. For convenience the table also contains the number of vehicles that were manually counted by me for each video. These are used to evaluate the performance of the algorithm and are displayed in the second row of the table labeled Ground truth. The last row of the table contains the error ratio of the algorithm for each of the video.

In addition to the table, I also built four graphs, one for each video. The graphs are much more practical than the table because they visually display the performance of the algorithm. Therfore, the classification results for the video 1 that was recorded in the morning at 9AM are shown in figure 49. The algorithm counted 176 small vehicles and 22 large vehicles. That is, it reported 12 small vehicles more and 3 large vehicles less than are in the video. The error ratio for small vehicles is only + 7.31%, and for large vehicles it is only -12%. These results are very good.

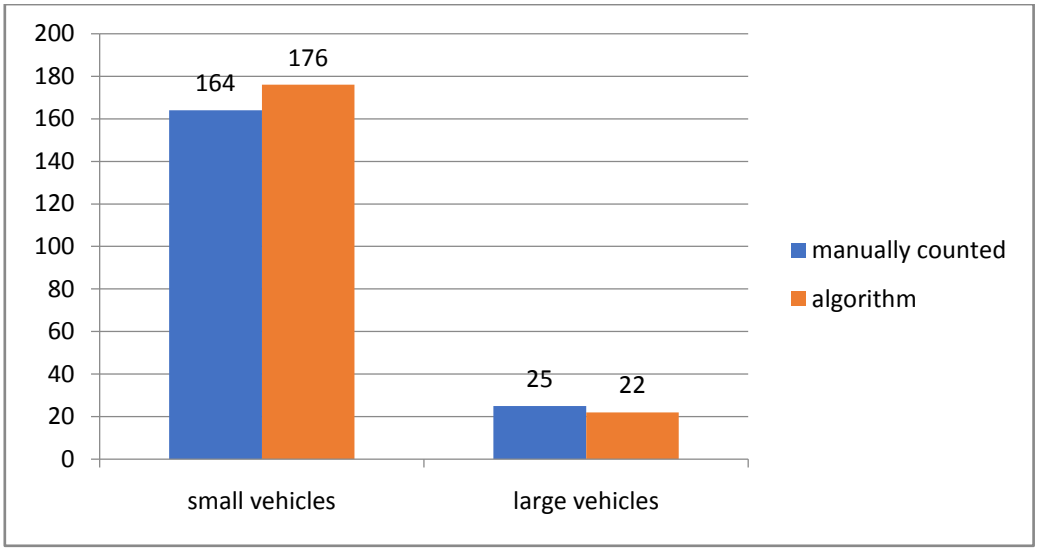


Figure 49 - Classification results and ground truth for the first video

For the video 2 that was recorded at 13h the classification results along with the ground truth are shown in figure 50. The algorithm reported 421 small vehicles and 21 large vehicles. That is, 14 small vehicles more and 3 large vehicles less than are in the video. The error ratio for small vehicles is +3.44% and for large vehicles is -12.5%. The higher value of the error ratio for large vehicles does not mean that the algorithm misclassified many large vehicles. It is larger because there are very few large vehicles in video and so each vehicle has a high weight in error rate.

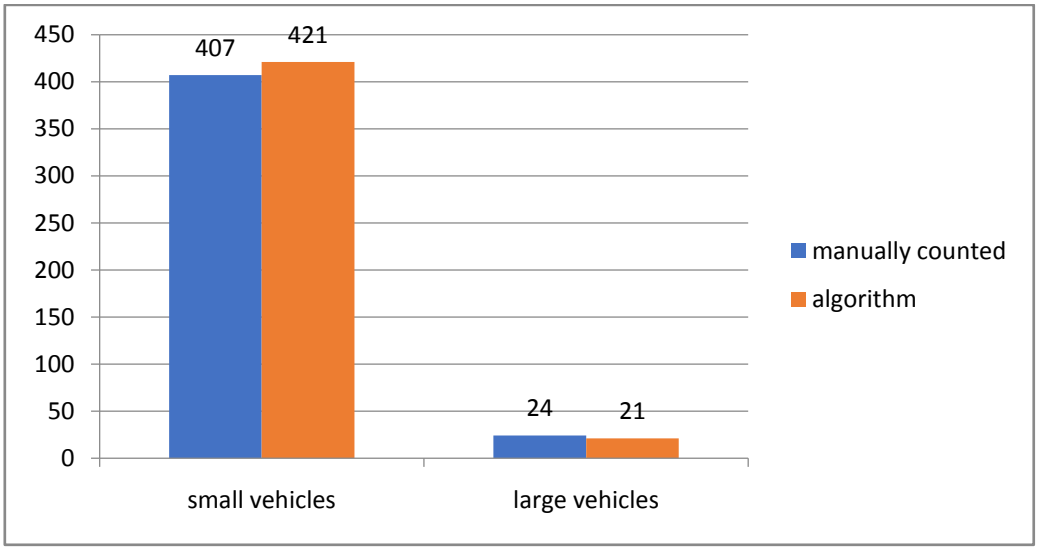


Figure 50 - Classification results and ground truth for the second video.

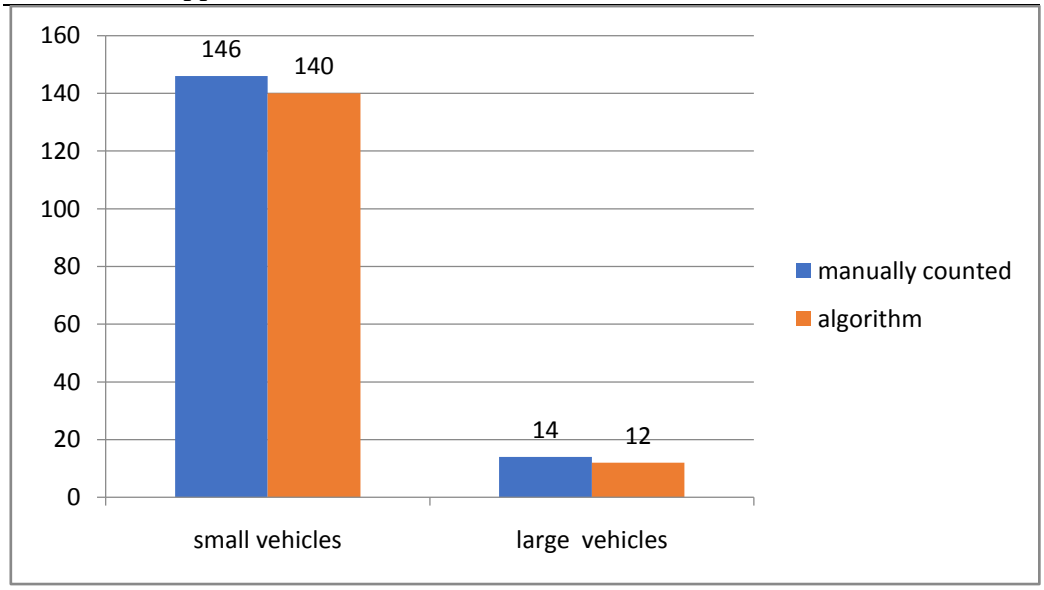


Figure 51 - Classification results and ground truth for the third video

Figure 51 shows the results for video 3, which was recorded at 18h. In this test scenario the algorithm detected 140 small vehicles and 12 large vehicles. That is, 6 small vehicles less and 2 large vehicles less than are in the video. These results are the closest to the ground truth compared to the results for the video 1 and video 2. The error ratio for small vehicles is only -4.11% and for large vehicles is only -14.29%.

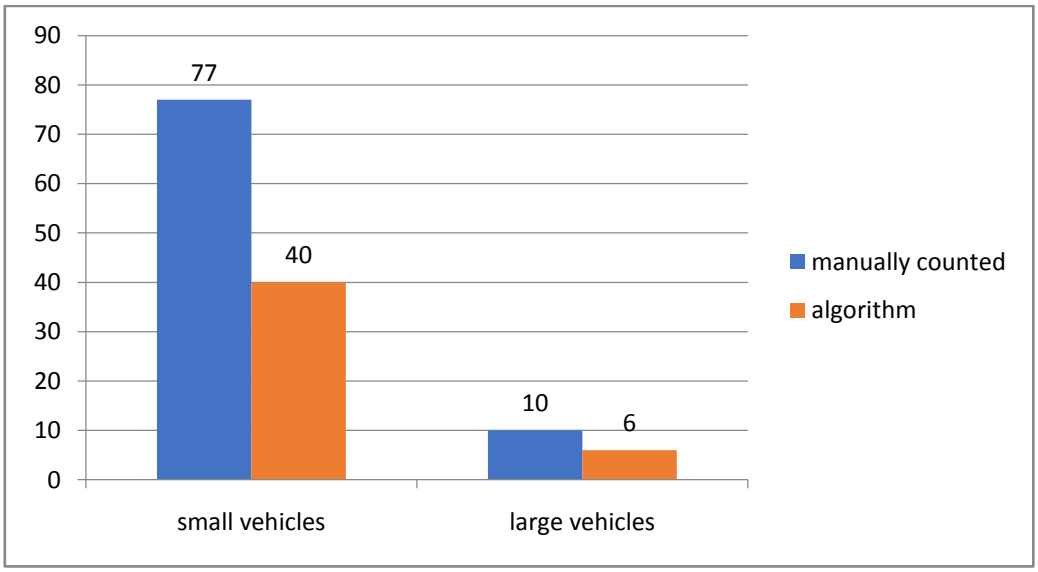


Figure 52 - Classification results and ground truth for the fourth video

The results for the video 4 are not so good. These are shown in the graph in the figure 52. The algorithm detected only 40 small vehicles out of 77 and only 6 large vehicles out of 10. In this test scenario the error ratio for small vehicles is –48.05% and for large vehicles is -40%. This lower detection performance of the algorithm is due to poor scene lighting. At the time of recording the video, that is at 11 pm, the intersection was artificially lit by street lamps. Under these conditions, the algorithm struggled to distinguish vehicles from objects in the background and thus failed to count some of the vehicles in the video.

With video 4 I finished testing the algorithm on the second data set. Of the four videos in the dataset, the algorithm detection performance was best for videos 1, 2, and 3. Although the results for these videos are close to the ground truth, they are not perfect. The algorithm failed to count a few large vehicles in each video. Also, for videos 1 and 2 the algorithm counted more small vehicles than are in the scene.

These counting errors are due to the imperfections of the background subtraction algorithm. Normally, a vehicle is represented by a big white blob in the foreground mask. But sometimes part of the vehicle's texture matches the texture of the scene and the background subtraction algorithm classifies this part of the vehicle as belonging to the backround. This phenomenon tricks the background subtraction algorithm.

So instead of representing the vehicle as one big blob in the foreground mask, the algorithm represents it by two or more smaler blobs. Therefore, a large vehicle or a small vehicle is perceived by the algorithm as two or more small vehicles. This problem makes the algorithm to count more small vehicles and less large vehicles. It is difficult to solve because the background subtraction algorithm can not make a diference between the background color and the vehicle color.

In video 4 is another problem for which the algorithm failed to get good results. Many of the vehicles were not recognized by the algorithm. This is because the background subtraction algorithm is not able to segment very well the foreground objects during night conditions. Unfortunately, to solve this issue, it is needed either to use the night vision camera or to increase the lighting in the intersection by adding more street lamps.

As a conclusion, I consider that the detection performance of the proposed algorithm for traffic surveillance is decent for monitoring vehicles in daylight conditions. In practice no better estimation is needed. The proposed video analysis solution is a good compromise between costs of deployment and accuracy. It has an excelent detection accuracy and it can run on low resource hardware.

6 A framework for behavior classification based on motion understanding

6.1 Introduction

Inspired by the success achieved by the algorithm for traffic surveillance I managed to design another computer vision algorithm. I published this algorithm in the article [BM2] and presented it in the first PhD report [BM5]. This novel approach is the continuation of the work presented in the previous chapter. That is, it is a computer vision algorithm that is meant to run on smart surveillance cameras in order to detect important events in the scene. The algorithm is meant to extend the capabilities of the surveillance system I propose in the figure 27.

Therefore, similar to the traffic monitoring algorithm, it is carefully designed. It uses low complexity computer vision algorithms that work in harmony in order to analyze complex video scenes. This approach allowed me to design an algorithm that has excellent detection accuracy and does not consume much computing power, so it can run on a smart surveillance camera.

The algorithm I propose in this chapter is more advanced than the first, because it is based on detecting the behavior of objects in the scene. It is aimed for increasing the public security in the big cities. Specifically, the algorithm is able to automatically detect a dog attack on humans by analyzing video images. It does not consume many resources, so it can run on the smart surveillance camera.Therefore, the algorithm enables the surveillance system to detect in real time dog attack events, which are for example very useful for the police. If the system alerts the police in real time that a dog attack is taking place, they can get to the crime scene very quickly to stop such dangerous or even in some cases deadly actions.

I believe that dog attacks in urban areas is an emerging issue that needs to be addressed as soon as possible.If you do a simple Google search for dog attacks, it will show you about 12,900,000 results. This is a huge number which demonstrates the seriousness of this problem.This finding motivated me to design a computer vision algorithm that can automatically detect such dangerous events. It is more advanced than the first one as it can detect behavior of dogs and humans.

Nevertheless, the proposed algorithm can run in parallel with the algorithm for traffic surveillance on a smart surveillance camera. In this way, the functionality of the system I propose in this manuscript is extended because each camera is able to detect two types of events. Every time the camera detects an event regarding traffic or attacks of dogs on people it immediately sends it to the control center.

Of course the algorithm does not depend on the traffic surveillance algorithm. They both work idependently. Therefore, the algorithm for detecting dog attack events can run on its own on a smart surveillance camera. Also, it is not designed to analyze only the scenes that contain traffic. It successfully detects the dog attack events in video footages that are captured from places in large cities where these types of events are more likely to happen like crowded urban streets.

Regarding the installation of the cameras, the proposed algorithm requires only two conditions to be fulfilled in order to function correctly. That is the camera must capture the scene from a side view and the surveilled zone must be close enough to the camera.Both conditions are due to the fact that the algorithm uses the silhouettes of objects in the classification process. As we all know, silhouettes are very sensitive to the angle of the camera. So in order to classify objects correctly, it is absolutely necessary for the algorithm to view the objects from the side. Otherwise, the silhouettes will be deformed and the algorithm will not be able to classify them.Similarly, if the monitored scene is too far from the camera, the analyzed objects are very small, and as such their silhouettes are useless for the algorithm.

The camera angle and the distance between camera and the surveilled zone are two mandatory requirements. The algorithm will not work at all if they are not met.My recommendation on the distance is that the observed area should not be more than twenty meters away from the surveillance camera for the algorithm to work properly.

To better understand why these requirements are mandatory and also to understand the principle of operation of the proposed attack behavior detection algorithm, in the following chapter, I will describe in detail all the building blocks of this algorithm.

6.2 Block diagram

The main components of the proposed attack behavior detection algorithm are illustrated in the block diagram in figure 53. I grouped the components into two groups. The low level processing group and the high level processing group.

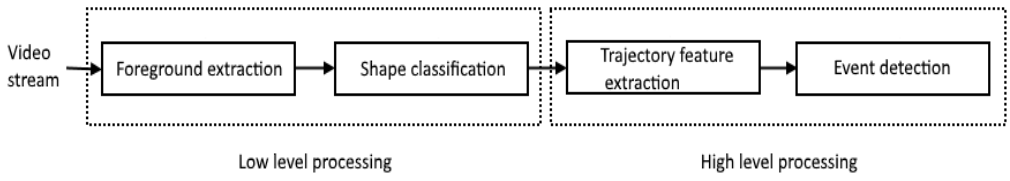


Figure 53 - Block diagram of the proposed attack behavior detection algorithm [BM2].

Basic video analysis operations such as foreground extraction and shape classification are performed by the first group.These algorithms are used to detect and classify foreground objects in the scene. Specifically, foreground extraction separates moving objects from static background, and shape classification identifies and classifies foreground objects.

The shape classification component is able to identify the humans and dogs in the scene. For each frame, it tracks the detected objects and saves their location, velocity and acceleration in the feature vector T_j. Because thesizeof feature vector is fixed, the objects are tracked only for N frames. To maintain the continuity, the content of the feature vector is updated with each new frame.

More complex operations are performed by the high level processing group.The algorithms used in this group are capable of recognizing the activities in the video with the help of trajectories.Trajectory feature extraction and event detection are the two operations involved in this process.

6.3 Low level processing

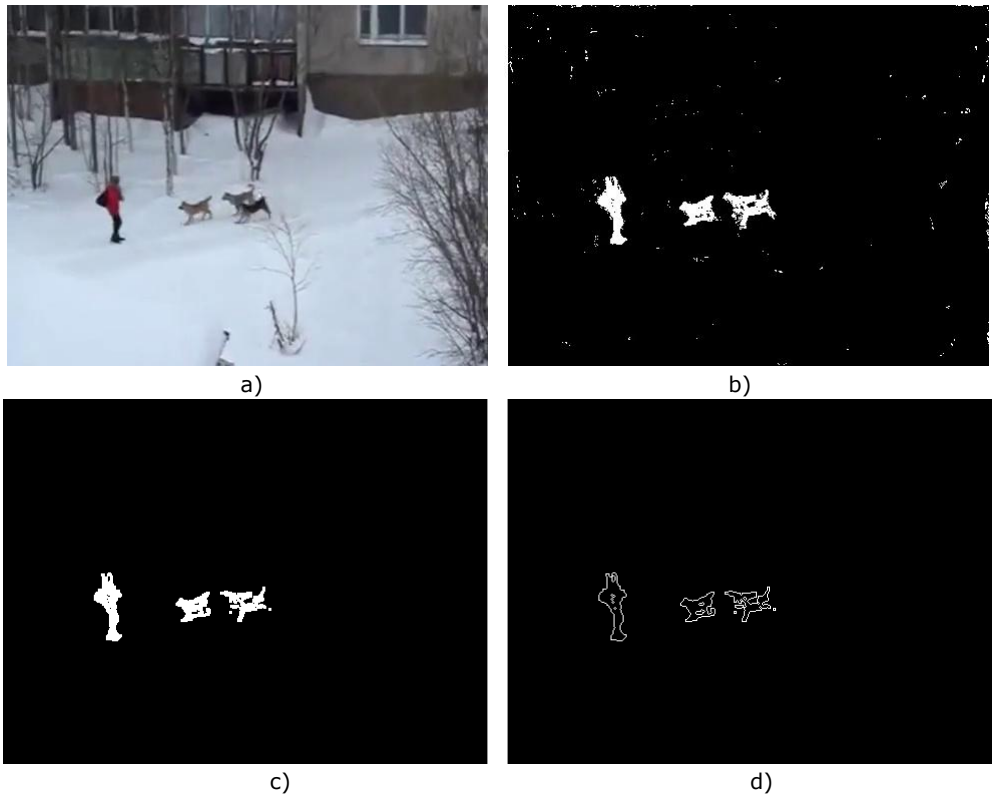
6.3.1 Foreground extraction

The foreground extraction phase includes a background subtraction algorithm and several image processing algorithms. Similar to the traffic surveillance application, described in the previous chapter, the foreground objects are extracted using the Mixture Of Gaussian (MOG) background subtraction algorithm [96]. Image a) in figure 54 shows a frame from a video that I use in my experiments, and image b) illustrates the foreground mask of the frame that is generated by the MOG algorithm.

In order to extract the foreground pixels, the mask is applied on the original frame. As a result, the newly generated foreground frame contains only groups of pixels that represent the foreground objects.But, beside of moving objects, the foreground frame also contains many pixels from the background. These noise pixels are due to the imperfection of the background subtraction algorithm and due to the camera noise.Usually they are single pixels, but in some cases the noise pixels can be grouped together. They appear as tiny blobs in the foreground mask.To fix this issue I apply the morphological opening operator on the mask.

The morphological opening is a composite operator. It is composed of an erosion and a dilation. First, the operator uses erosion to remove the noise pixels. Next, to remove the mask deformations introduced by the erosion, it applies the dilation to the eroded mask. Figure 54.b illustrates the mask before processing, and figure 54.c

illustrates the foreground mask after processing. The opening operator uses the same kernel for both erosion and dilation. In the current application, this kernel has been set to a rectangular shape of size 3x3.



a) b) c) d)

Figure 54 - a) Original frame; b) Foreground mask; c) Foreground mask after morphological opening; d) Object contours

The objects in the scene are segmented according to the same algorithm used in the traffic surveillance application. The border following technique, proposed by Suzuki and Abe [90], successfully generate the contours of humans and dogs (figure 54.d).

During the experiments, I noticed that in certain cases, the objects are far away from the surveillance camera. In these situations, their features such as legs and arms are extremely small. The noise of the camera also affects the objects. It further distorts the already small features. Under these conditions, it is practically impossible to identify any distant object in the scene.

As a solution to this problem, I propose the use of a filtering procedure. Its purpose is to disregard small objects. The procedure consists in comparing the contour area of the object with a threshold. The object is disregarded if its contour area is smaller

that five houndred pixels. After applying this filtering procedure, the resulting foreground mask contains only the objects of interest.

6.3.2 Shape classification

The process of classifying the shape of objects was another challenge that I faced in the process of designing the proposed algorithm for detecting attack behavior. I considered several shape classification algorithms from the literature.

An interesting algorithm for shape classification I investigated was proposed in [97].The algorithm is based on the idea of shape decomposition. It defines the shape as having strand and base structures. In the process the algorithm uses a set of shape templates. Each element in the template set is represented by its base and strand structures.

The algorithm at first decomposes the input shape. Then it tries to find a match by comparing the strand and base structures of the input shape with the base and strand structures of the elements in the template set. As you may have guessed, the algorithm recognizes the input shape if a match is found. The disadvantage of this approach is that it is very sensitive to shape deformations. If the shape of the object is distorted even with a small amount, the algorithm will not recognize it. Due to this shortcoming, I decided not to use it in my application.

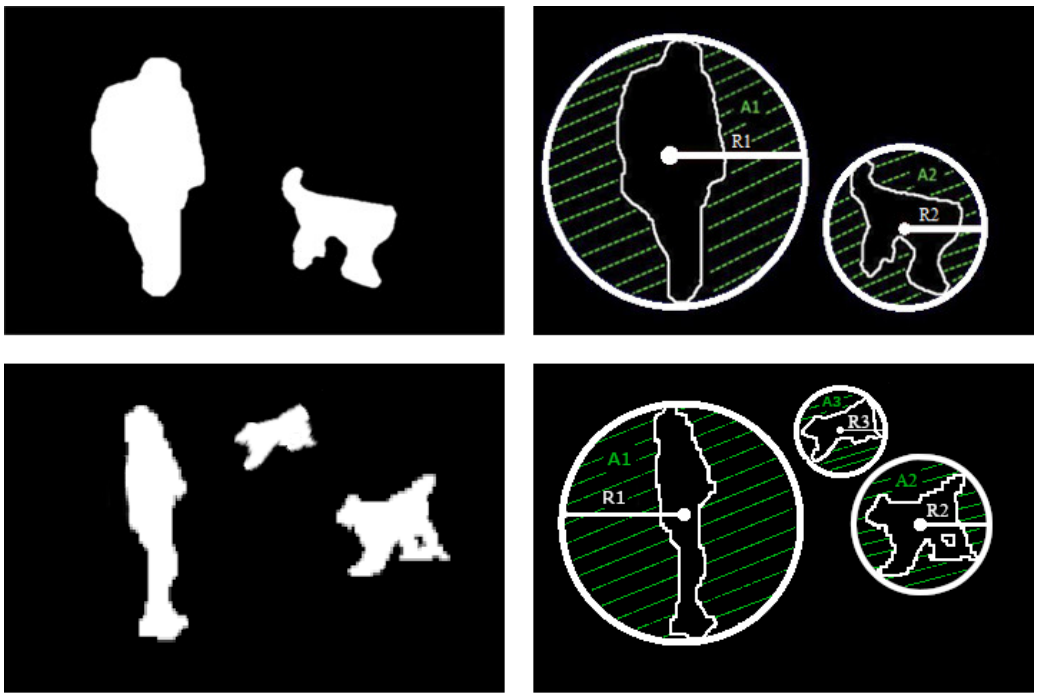
Another solution that appeared appealing to me was the shape classification algorithm presented in [98]. It works in the same manner as the algorithm described in [97]. The solution uses the template matching technique. But, instead of decomposing the shape into strand and base structures, it uses the object's skeleton. The algorithm thins the blob of the object until a skeleton is obtained. However, my first attempt to use this shape classification algorithm was a failure. Many objects were not correctly identified. Usually, the algorithm fails to identify an object if its skeleton is not nearly perfect.This is due to the fact that even a small change in shape results in a rather large change in the skeleton of the object.

The urban areas are mostly crowded. Many object shapes are deformed due to occlusions. Therefore, it was no surprise that the algorithms proposed in [97] and [98] did not work. Based on the knowledge gained from the experiments, I managed to successfully develop a novel shape classification algorithm. The classifier is based on the following features:

- Rbound; blob bounding circle radius
- ΔS = circle area – blob area; area difference

The algorithm calculates minimum bounding circle of each blob in the foreground mask. Figure 55.a illustrates a simplified foreground mask that contains only a human and a dog. The bounding circles corresponding to these objects are illustrated in figure 55.b. You can see that object sizes are successfully captured by the radii R1 and R2 of the circles. A simple comparison of these radiuses can tell us whether the silhouette is of a human or a dog. The radius of the human silhouette is much larger than the radius of the dog's silhouette.

At first glance it may seem that this approach will work well. Still, experiments have revealed that, in certain cases it fails. Sometimes, the silhouettes are the same size but they represent different objects.



a) b)

Figure 55 - a) A simple foreground mask that contains only a human and a dog; b) The objects features. The radii and areas of the minimum bounding circles [BM2].

To solve this shortcoming, I have designed an additional shape feature that I use in the proposed classification algorithm. The area difference ΔS, that is the bounding circle area minus the blob area, provides additional information about the blob shape. An illustration of this principle is shown in figure 55.b. The A1 is the ΔS_H of a human and A2 is the ΔS_D of a dog. The area difference of the human siluet is greater that the area difference of the dog silhouette $\Delta S_H > \Delta S_D$.

The proposed classification algorithm uses two thresholds, T_{radius} and T_{area}. An object is identified as belonging to the human class if the radius of its bounding circle R is greater than the threshold T_{radius} and its area difference ΔS is greater than the threshold T_{area}. Otherwise, if the R is smaller than T_{radius} and the ΔS is smaller than T_{area} the object is identified as dog. If the object does not meet the requirements to be classified as a dog or human, it is labeled as belonging to the other class.

In order for the classification to work properly, the paramers T_{radius} and T_{area} of the proposed algorithm should be carefully set. They depend on the environment, therefore their values must be established experimentally. For this application I discovered that the T_{radius} set to two centimeters and the T_{area}set to eight houndred pixels give the best results.

Both, the radius of minimum bounding circle and the area difference are robust features. They tolerate small deformations of the shape. The proposed shape classification algorithm has a high detection rate, and it can be easily modified to recognize other types of objects. In this application there was no need for many classes. Dog attacks usually take place in places that do not contain vehicles or other types of objects other than humans and dogs.

6.4 High level processing

6.4.1 Trajectory and movement analysis

In order to capture the attack of dogs on humans it was necessary to understand the nature of such events. Usually the attack is carried out by a group of dogs and not by a single dog. In the beginning phase, the dogs observe the victim and begin to form a group. After that, they start to run in the group until they reach the target. To capture this event, I propose the use of blob trajectories and the speed of the blob as features in the detection process.

Trajectory analysis

Trajectories can reveal us significant information about dog attacks. By carefully analyzing them, it is possible to predict such dangerous events. The idea relies on the fact that the trajectories of dogs aiming to attack a human converge to a single place in the scene.

Therefore, by testing the trajectories of dogs running for convergence, you may find out if the scene contains attack actions. Usually, all the dogs in the scene are involved in the attack. Of course, this is not always true. In very rare situations, not all of them participate in the attack. Sometimes one or two dogs do not engage in this action. So, the analysis of all the trajectories of the dogs is not suitable for such

a case. To deal with this situation as well, the proposed algorithm analyses the trajectories of the crowd and ignores the trajectories of isolated objects.

According totheseobservations, the trajectories of dogs can be either convergent or divergent. If the trajectories converge to a point in the scene, there is a high chance that an attack will occur. On the other hand, if the trajectories are divergent, this means that the group of dogs breaks and that there is very little chance that the dogs will attack the victim.

Movement analysis

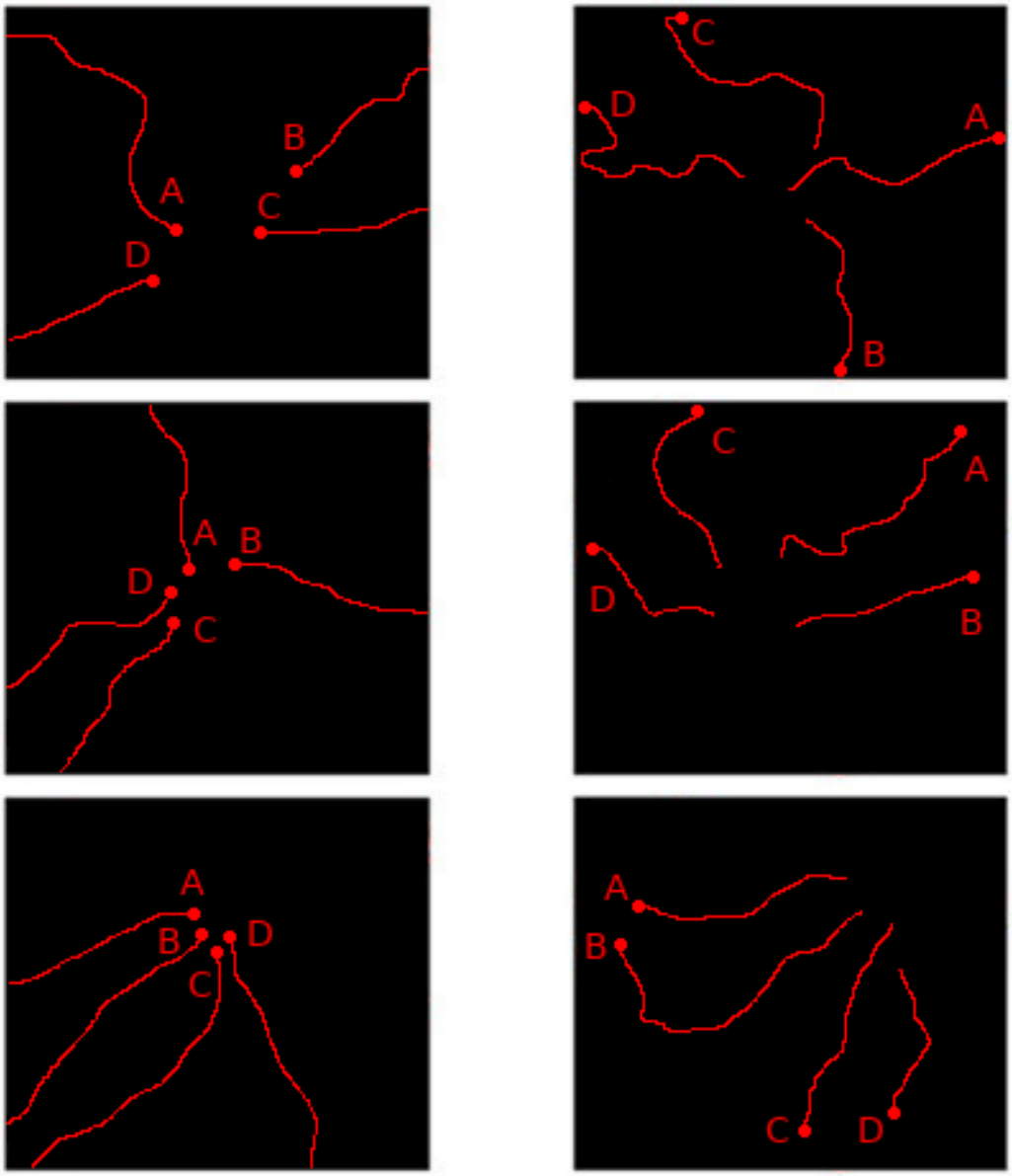
In order to improve the detection accuracy of the proposed attack behavior detection algorithm, I decided to include a new feature in the process. My belief is that the velocities and accelerations of the dogs will improve the detection accuracy. These features can accurately reveal attack phases.

In a typical attack scenario, the accelerations and velocities change as follows:

- acceleration > 0 – indicates the beginning phase
- acceleration = 0 and velocity > 0 – this is the intermediary phase
- acceleration < 0 – denotes the final phase

The attack begins when the dogs see the victim and start running towards her. This is the time when the accelerations of all dogs reach their highest values. Also, the velocity of each of them starts to increase gradually.

Soon after the beginning phase, the accelerations will start to decrease until they are zero. This indicates the start of the intermediary phase. The dogs have set their maximum speed and are on their way to the victim. This phase is the longest one. Certainly, there are some exceptions, such as when the distance between the group and the victim is very short. In this scenario, dogs do not have enough time to reach their maximum speed. Therefore, the intermediary phase may be very short or, in some cases, may disappear.



a) b)

Figure 56 - a) Convergent trajectories; b) Divergent trajectories [84]

After the intermediary phase follows the final phase. At this stage the group is very close to the victim. This phase begins when the individuals in the group begin to decelerate, that is their accelerations are negative and the velocities begin to decrease. Finally, when the group reaches the victim, their velocities and

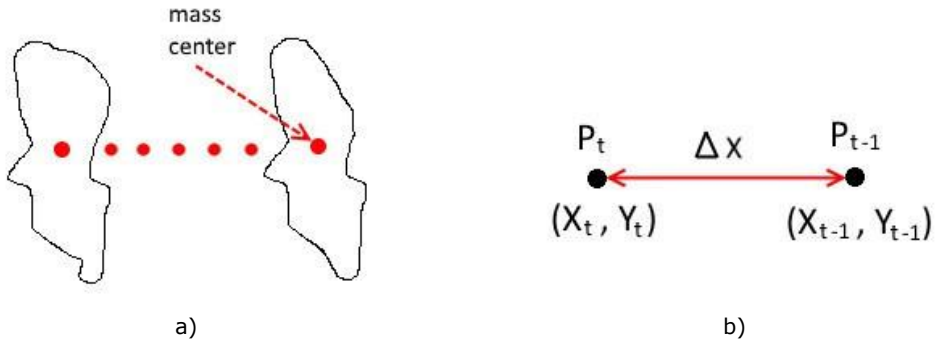
accelerations are all zero. The dog trajectories of such scenarios are illustrated in the figure 56. The image in a) shows the trajectories that correspond to the attack action. In this case all the trajectories converge towards the victim position. In contrast, the trajectories from the image b) correspond to the non attack scenarios. That is, the dogs are not interested to attack the victim, so they randomly walk in the scene. In this scenario the trajectories of dogs are all divergent.

6.4.2 Trajectory feature extraction

According to the hypothesis described in Sections A and B, two trajectory characteristics are required for event classification. The trajectory points describe the path of each dog in the scene. While the speed of the path points helps the algorithm to capture the phases of the attack.

To find the path of the object, the proposed algorithm uses the vector and positions of the object. The position is calculated in each frame according to the following principle. Once the contour of an object is available, the algorithm calculates its center of mass using the image moments. The mass center P(x,y) illustrated in figure 57.a is actually the position of the object in the current frame. By storing the object positions for multiple frames the proposed algorithm is able to compute the trajectory of a dog in the scene. The same procedure applies to all dogs on the scene.

Since the trajectories are not enough to detect the attack, the proposed algorithm calculates the speed and the acceleration of each dog in each frame. These two features can be easily obtained from the trajectory points. Two successive trajectory points P_{t-1} and P_t (figure 57.b) of the object indicate how far he moved from time *t-1*to time *t*. This distance is measured by counting the number of pixels between the position of the object at time t-1 and time t.



a) b)

Figure 57 - a) Blob mass center; b) Distance between two successive blob points

To this point we know how far the object moved and the elapsed time needed for this action. Therefore, its velocity can be easily computed by using the equation (13).

$$v_t = (P_t - P_{t-1})/dt \tag{13}$$

The acceleration of the object is defined as velocity change in a time interval. So the acceleration of the object is obtained by dividing the velocity difference v_t -v_{t-1} with the elapsed time from the moment *t-1* to the moment *t* as in equation (14).

$$a_t = (v_t - v_{t-1})/dt \tag{14}$$

The information regarding the object is stored in the feature vector T_j(15). This vector contains all the object data that is collected in a video frame. The position is represented by the coordinates x_j and y_j, the velocity by v_j and the acceleration by a_j.

$$T_j = x_j , y_j , v_j , a_j \tag{15}$$

The trajectory feature of the objectT (16) is defined as a collection of such feature vectors. It consists of a group of feature vectors extracted from N video frames.

$$T = [T_1 , T_2 , \dots , T_N] \tag{16}$$

The trajectory feature T captures both the static and dynamic feature of the object in a time-frame. Each dog in the scene has a trajectory feature vector.

6.4.3 Event detection

The data stored in the trajectory feature vector T is further used to classify the video frames. A video frame is classified as belonging to normal or dangerous class. It cannot belong to both classes in the same time and it cannot be left unclassified. For the classification task I chose to use the (Support Machine Vector) SVM algorithm. The algorithm has been used successfully in similar applications for event detection [99], [100]. Therefore, after a few experiments we became convinced that this algorithm will be able to classify attacks based only on the feature vector.

For each video frame, the feature vector Tj is calculated. The vector is saved in a circular buffer that is three hundred in size.This principle allows the storage of the trajectory features of objects for a time interval of about twelve seconds. Because the event cannot be accurately captured by one or several frames, the algorithm waits until the circular buffer is full and only then calls the classifier. When this happens, all data in the circular buffer is transmitted to the input of the SVM algorithm.

The qualities of the SVM classifier are that it has a low computational complexity and good classification performance. During the experiments I noticed that, even if it was trained with a small number of samples, the algorithm managed to classify the data correctly. The algorithm uses support vectors in the decision making process, therefore the result is very little influenced by class distributions. This feature allows the classifier to have good classification performance and high processing speed.

6.5 The dataset

The dataset used for testing the proposed dog attack detection algorithm is made of a set of 119 video clips collected from public internet sources [101], [102]. Samples of several video clips are illustrated in figure 58. All of them contain at least one dog attack scene. The surveillance camera with which the videos were recorded was positioned so as to capture all the objects in the scenes from the side.

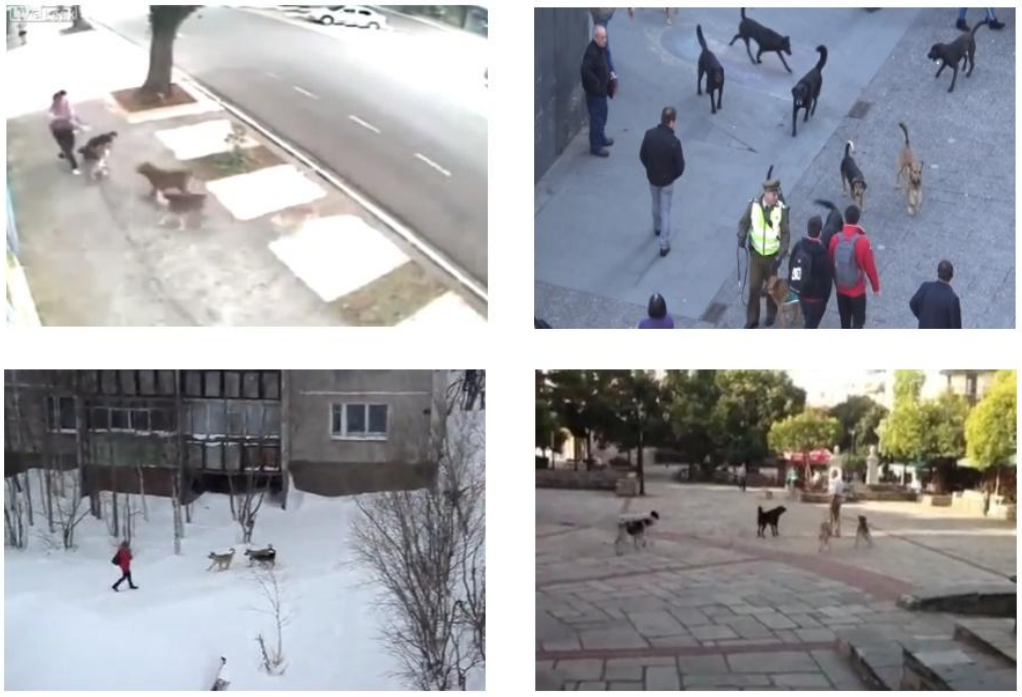


Figure 58 - Samples of the dataset used to test the proposed algorithm for detecting dog attack

6.6 Experiments and results

To assess the performance of the proposed attack event detection algorithm we designed a software prototype. The program was written in C++ programming language and uses the basic algorithms defined in the openCV computer vision library. To save time, we decided to take advantage of this library and not write the basic image processing algorithms from scratch.

The validation software consists of the feature extraction part and the classification part. As you may already have noticed the prototype software written in C++ is responsible of extracting the features. After the video is processes the software saves the generated features in a Weka Attribute-Relation File Format (ARFF). The classification is done using the Weka framework. This is why the output follows the ARFF file format.

We used the Weka framework for the classification part because it offers the SVM classifier and is relatively easy to use. In order to be able to classify attack events, the classifier had to be trained before use. For this purpose, I selected a set of 89 videos from the public Internet [101],[102]. The training videos were chosen very carefully, so as not to contain inappropriate camera angles or unfavorable weather conditions, such as rain or fog.

In testing phase, we selected another 30 videos, other than those used for training. After processing, the prototype program generated a total of 503 trajectories. 230 of them were the trajectories of the videos that contained a dog attack, and the rest were of the videos that did not contain an attack event.

The SVM correctly detected the attack events. It also classified right almost all normal events. Only a small portion of normal events were miss classified as shown in the confusion matrix in Table 9. Out of a total of 273 normal events, the algorithm correctly detected 268 and missed only 5 events. Thus, the SVM algorithm classified 99% of the trajectories correctly and only 0.99% of the trajectories incorrectly.

	Predicted		
		attack	normal
Actual class	attack	230 TP	0FN
	normal	5 FP	268 TN

Table 9 - Confusion matrix [BM2].

Experiments and results

From the confusion matrix, the following performance assessment parameters were evaluated:

True positive rate (17) - Sensitivity:

$$TPR = \frac{TP}{TP + FN} = \frac{230}{230 + 0} = 1 \tag{17}$$

True negative rate (18) - Specificity:

$$TNR = \frac{TN}{TN + FP} = \frac{268}{268 + 5} = \frac{268}{273} = 0.98 \tag{18}$$

False positive rate (19):

$$FPR = \frac{FP}{FP + TN} = 1 - TNR = 1 - 0.98 = 0.02 \tag{19}$$

False negative rate (20):

$$FNR = \frac{FN}{FN + TP} = 1 - TPR = 1 - 1 = 0 \tag{20}$$

Performance parameters show us that the algorithm was able to accurately detect events in the scene. It has a true positive rate (sensitivity) equal to one which means that it correctly detected all dog attacks. Hence, in this testing case, none of the attacks were classified as belonging to the normal class.

Regarding the normal class detections, the score is similar to the attack class detections. The true negative rate (specificity) reveals us that the algorithm classified almost all normal scenarios correctly. It classified only five normal sequences wrongly, while the remaining two hundred sixty-eight sequences were classified correctly. Similarly, the false negative rate and the false positive rate performance indicators tell us the same think. That is, the algorithm did not classify any attack event as belonging to the normal class and it classified only two percent of the normal eventsas belonging to the attack events.

Given these results, I can conclude that the proposed algorithm for detecting dog attacks is excellent for capturing such dangerous events in urban areas. If used properly it can surely improve the security in the smart cities by automatically analyzing the videos which are captured by surveillance cameras. This fact will ease the work of the surveillance officers and improve the efficiency of the whole surveillance network. Hence, the security in the urban areas which uses such a system will also be improved.

Another benefit of the proposed algorithm is that it can be easily adapted to recognize new events. It uses trajectories in the analysis process so this makes it more flexible to adaptations. That is, it can be easily adapted to recognize other events of interest, such as human fights, robberies, loitering detection at the airport, etc. I believe that the detection results of the novel versions of the algorithm will still be very close to the results presented in this chapter.

7 Proposed hybrid Deep learning/VA features solution for complex behavior classification in sensors environments

7.1 Introduction

In this chapter I am going to present another computer vison algorithm that is aimed to extend the functionality of the surveillance system illustrated in Figure 27. I published this algorithm in articles [BM3], [BM4] and presented it in the second PhD report [BM6]. Similar to the traffic analysis algorithm and the dog attack behavior detection algorithm that I proposed in the previous chapters, the algorithm in this chapter has also been carefully designed to be able to run on a smart surveillance camera. Moreover, it also pursues the same goal as the algorithm for detecting dog behaviors. That is, the algorithm is meant to increase the public safety in the big cities.

As already mentioned the algorithm does not require high computing resources to perform this task, so it can run on a smart surveillance camera along with the traffic analysis algorithm and the dog attack behavior detection algorithm. This means that the algorithm extends the functionality of the system proposed and illustrated in Figure 27. In addition to traffic analysis and detection of dog attacks, the system is now able to detect human fighting events in video. This is very useful for the public safety. If, for instance, the detected events are used to alert the police about the fight, the fighting action can be stopped, because the police will arrive on the spot very quickly. In this way, the proposed algorithm for detecting human fighting events will considerably increase public safety in these areas.

I believe that this approach to detecting violence in big cities will have a positive impact on the quality of life, because it makes cities safer. Thus, citizens living in this environment will feel safer. It is important to note that the detection of violence in large cities is an important issue. In the last decade, big cities are becoming overcrowded as more and more people migrate from rural to urban areas. This phonemenon increases the number of violence cases. So it is very momentous to

take steps to reduce these unpleasant events in order to increase the quality of life in these areas. This fact motivated me to design a computer vision algorithm for detecting fight events.

The algorithm is designed very carefully, so that it will successfully analyze almost any type of scenes. For instance, it is able to analyze the scenes that besides pedestrians also contain traffic lanes, as well the scenes that contain only pedestrians such as the streets in a city. In addition to this feature, the algorithm does not have any dependences, it can run alone or in parallel with other similar computer vision algorithms on a smart surveillance camera. For instance, the algorithm can run in parallel with either the traffic analysis algorithm, or the dog attack detection algorithm, or them both. This modular design makes the surveillance system flexible. That is, extending or reducing the functionality of the system is easy. The algorithms running on the camera do not depend on each other. So one can be added or removed on demand without affecting the algoritms that are already running on the camera.

In order for the algorithm to work properly the camera must be installed correctly. The requirements regarding smart surveillance camera installation are the same as for the algorithm for detecting dog attacks. That is, the camera should be installed so as to capture people from the side and the distance between the camera and the monitored area should be about twenty meters. This would be the setting for maximum detection performance. But, because the algorithm is robust, I am convinced that it will work correctly even if this distance does not fall within the imposed limit and the camera does not perfectly capture people from the side.

Neverthless, the work presented in this chapter refers to an effective network-based surveillance system capable of detecting human fighting events in urban areas. The system consists of low-performance nodes. That is smart surveillance cameras that are connected wirelessly to a control center, as shown in the Figure 27. Nodes are in charge for obtaining a maximum recall in capturing violent events. Hence, the goal is to catch as many as possible violent events and at the same time reduce the false alarm rate.

The benefit of such an approach is that only video data that is detected as violent is transmitted to the control centre for additional processing. In this way, the proposed system facilitates the work of surveillance officers, as they are not required to inspect video recordings continuously. They only need to inspect certain portions of the video that are reported by the system. Another advantage is that the system increases the detection rate of events, because it eliminates the human error generated by the "tired eye" syndrome from the process.

An important aspect of the novel violence detection algorithm described in this chapter is that it only uses a motion field feature and does not take into account the

appearance. Its block scheme is illustrated in the figure 59. The algorithm consists of a motion feature extractor, a deep neural network and a temporal filter.

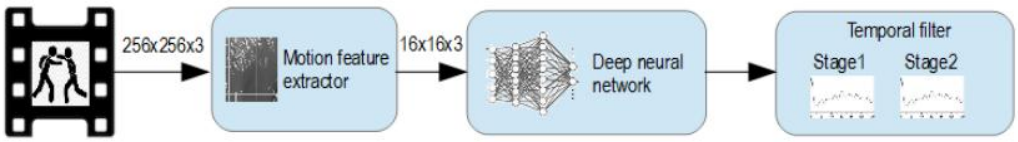


Figure 59 - Block diagram of the proposed surveillance system

The reason why I used only a motion feature for the fight detection algorithm was influenced by the experimental evidence on the recognition of human action presented in the paper [103]. It proved that the motion is able to capture more details than the appearance. According to the experimental results carried in this paper, in order for the appearance information to be used efficiently in the action classification process, large data volumes and sophisticated models are needed.

On the other hand, the approach that uses only motion features is completely invariant to appearance changes and does not require large volumes of data and sophisticated models. Moreover, the motion information remains stable even when ambient light changes. Thus, this allows the system to function successfully even in the night.

Another crucial choice in the design phase of the algorithm was the selection of the motion descriptor. I have to choose between MPEG flow and optical flow. According to the study conducted in the paper [104], the efficiency of MPEG motion vectors in recognition of human action is very high. Experimental results show that performance in detecting human action is close, and even better in some cases, to the performance of the optical flow based method.

In terms of computational complexity, the MPEG flow hugely outperforms the optical flow. The calculation of optical flow consumes a lot of resources, while MPEG flow can easily be obtained from encoded video stream without the need for major processing power. Moreover, the MPEG flow vector operates on blocks of size 16x16, thus for each block it provides only one motion vector. This means that the output generated by the MPEG for a video frame is small so that the input size required for the CNN classifier is greatly reduced. Although small in size, this motion descriptor still retains enough detail to discriminate actions in video.

Motivated by this finding I was convinced that the best choise for my application is the MPEG flow motion descriptor. But despite the fact that this handcrafted feature is used as classifier input, the network still needs to discover new features in the

MPEG flow vector field. This process forces the network to capture spatio-temporal nature of the motion patterns.

Nevertheless, in addition to selecting the motion descriptor, a key decision in the architecture configuration process was the selection of a cascade model for linking time domain with spatial domain. Most DL based action recognition systems use as inputs two streams, one for appearance and one for motion information. Such approach was not suitable for my application because it require large size architectures.

Therefore, in the spirit of the divide and conquer philosophy, I divided the fight event detection task between two processing cascaded modules. The first one is the CNN itself that receives the motion information for each frame from the video decoder. Its role is to output the likelihood of the presence of a fight action in a frame based on the spatial distribution of the motion within that frame. The role of the CNN in the system can be interpreted as a violence feature extractor, using a motion field input.

To make the final prediction regarding the presence of the human fighting event in a larger group of frames, the output of the CNN is used subsequently by a time domain processing module called the time domain filter. This module is the second one and is much simpler than the first one. It has scalar and binary inputs and so its computational burden is insignificantly small. Also I decided to use the cascade structure for this module, because it is easier to configure it this way.

The time complexity of the proposed algorithm for human fight event detection that I propose in this chapter is O(NxNxT). It is directly proportional to the size of the CNN input and the number of frames used in the classification process.

7.2 The Video Processing Unit

The ability of DNN to learn the features itself seems very attractive, so that skilled work, which otherwise requires specific application knowledge, to design features can be replaced by this approach. But this comes at a price. The qualified work needed for crafting the features is replaced by the process of data labeling. Sometimes, the labeling process can take advantage of publicly available datasets that are already labeled and prepared for training the convolutional network. Sadly, not all applications can benefit from this advantage because available datasets are limited to just a few domains.

Surveillance videos containing violence are very rare. It is almost impossible to find a large labeled dataset that encompasses this behavior. Indeed, there are a lot of small surveillance videos capturing violence in urban areas but they are not appropriate for my work. Based on research, I found that only the BEHAVE [105]

and ARENA [106] dataseets meet the requirements. Since they contain violent video clips that are short, I decided to use both of them to demonstrate the concept proposed in this thesis.

The purpose of this work is to combine the potential of a deep convolutianal network with the capability of carefully handcrafted motion features. To make learning possible by using a relatively small volume of data, the deep convolutional network is powered by a compact motion feature called MPEG flow vectors. As already mentioned in the previous chapter this feature encodes the motion that belongs to the 16x16 frame blocks.

Due to the efficient distribution in time and space, a wide variety of human movements can be captured by these low level features such as walking, jumping and running. To extract high level features from MPEG flow vectors, like fight and no fight events, the proposed approach combines a DNN and a time domain filter. More details about the filter are described in the subsequent section.

To reduce processing time, the proposed approach benefits from a small number of convolutional layers. At first glance, it may seem that the small data set causes lower performance. Because of the carefully tailored architecture, the algorithm presented in this work attains state of the art performance even if the training samples are small in number. Hence, I managed to design a modular distributed system capable of capturing violent events in urban areas.

7.3 Low Level Feature Extraction

The algorithm I propose in this chapter uses MPEG flow [104] motion features that are extracted by the video codec. The features are estimates of 16x16 image micro-block motions which are used by the video encoder to compress the data. According to the results published in [103],[104], MPEG flow is an efficient feature for capturing human motion in video. The authors of [103] use it with a bag of words model, whilst the paper [104] uses the feature with a spatiotemporal convolutional neural network.

Pixel movement in the 16x16 micro-block is estimated by exactly one motion vector. Its computation is made by using the position of the current micro-block and the best matching position in the previous frame, as illustrated in Figure 60. The yellow rectangle in the left image shows the position of a micro-block in the current frame, while the green rectangle in the right image shows the position of the block in the previous frame. The motion vector, representing the movement of the block, is shown by the red arrow in the right image. It is represented by the video encoder through two reference points. The first corresponds to the position of the block in

the current frame and the second is the position in the previous frame as stated in equation (21).

$$\mathbf{v}_m = \mathbf{p}_{source} - \mathbf{p}_{destination} \tag{21}$$

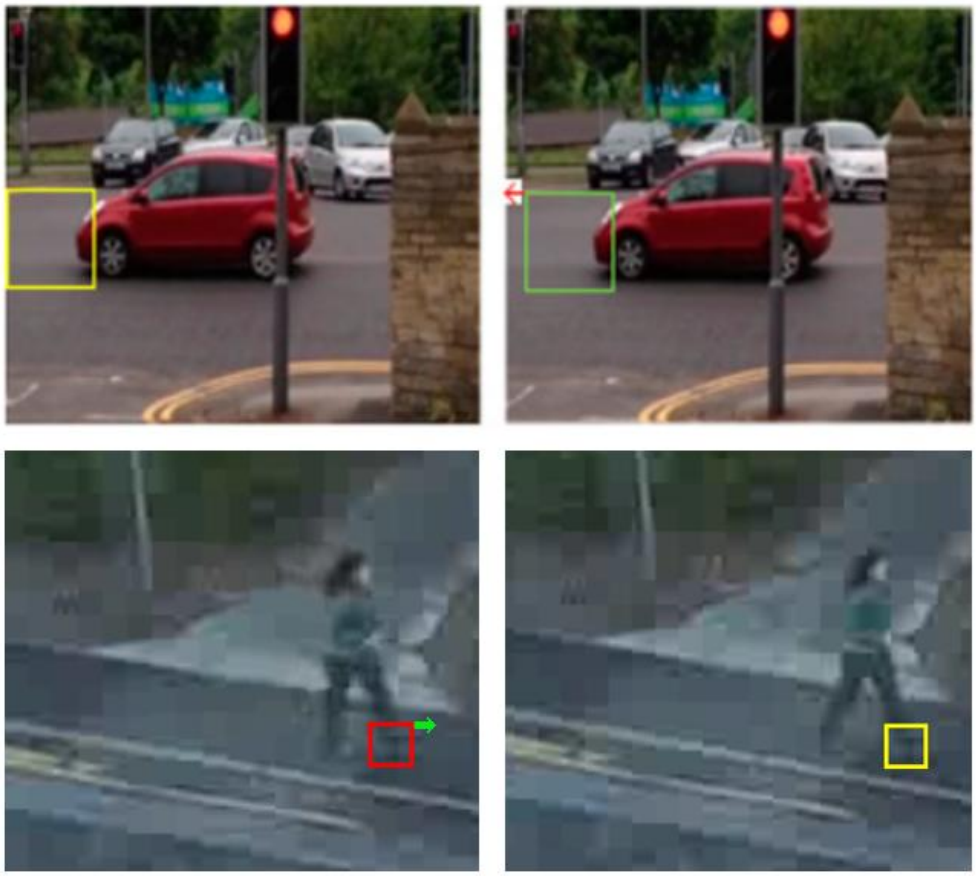


Figure 60 - Motion estimation rule; The image on the upper left illustrates a block in the current frame, the yellow rectangle; The image on the upper right illustrates its best fit in the previous frame, the green rectangle and the red arrow shows the motion vector. In a similar manner, the yellow rectangle in the bootom right image illustrates a block in the current frame and the red rectangle in the bottom left image illustrates its best fit in the previous frame. Note that the blocks are not in real size, it's just for the illustration purpose.

For discovering the best match of the micro block, the video encoder takes advantage of a distortion measure called the sum of absolute differences SAD(v) defined by equation (22).

$$SAD(\boldsymbol{v}) = \sum_{p} D(\boldsymbol{p}, \boldsymbol{v}) \tag{22}$$

Parameter **p** from the equation is the position vector corresponding to the pixel in the current micro-block, and parameter **v** is the approximate pixel motion vector. Equation (23) specifies the distortion measure used in the estimation process.

$$D(\boldsymbol{p},\boldsymbol{v}) = |Img(\boldsymbol{p}) - Ref(\boldsymbol{p}+\boldsymbol{v})| \tag{23}$$

The variable *Img* (**b**) is the pixel with position **b** in the current micro-block. *Ref* (**b + v**) denotes a pixel with position **b** in the micro-block displaced by position vector **v** in previous frame. Computation of the distortion measure of the whole block is given by the Equation (24).

$$SAD(\boldsymbol{v}) = \sum_{p} D(\boldsymbol{p},\boldsymbol{v}) = \sum_{p} |Img(\boldsymbol{p}) - Ref(\boldsymbol{p}+\boldsymbol{v})| \tag{24}$$

In comparison with the more traditional L2 norm, the L1 norm used in the SAD equation is less sensitive to the presence of outliers in the data [107].

Computation of the MPEG flow features is very fast. Due to the hardware support for the video decoding process, the cost of calculating it in terms of resources used is low. For each frame, the decoder produces a list of motion features. These are calculated at block level and only the 16x16 blocks showing movements are encoded in the motion feature extraction process.

Each element of the extracted list contains a pair of source and destination position vectors of a 16x16 micro block. Subsequently, the motion vectors are obtained by processing all the elements of the list. To eliminate the noise introduced by the surveillance camera and small objects that are not of interest for this application, I applied a threshold procedure on each vector v_m in the list. In this way the vector v_{mth} (25) is set to zero whenever the original magnitude is small:

$$\mathbf{v}_{mth} = \begin{cases} \mathbf{v}_m, \text{if } |\mathbf{v}_m| > minMotionTh \\ 0, otherwise \end{cases} \tag{25}$$

The main steps of the algorithm used for estimating the MPEG flow motion magnitude are presented in the following pseudo code.

```
1:  Extract the set ξ of motion vectors from the MPEG stream
2:  Create an empty array ζ to store the motion on X and Y axes
3:  FOR each motion vector V in ξ
4:  DO
5:    Get the source and the destination points of the motion vector V
6:    Compute motion magnitude of V
7:    IF magnitude of V > minMontionTh THEN
8:        Compute the x motion corresponding to the X axis and add nCorrection
9:        Store x into ζ on corresponding plane
10:       Compute the y motion corresponding to the Y axis and add nCorrection
11:       Store y into ζ on corresponding plane
12: END
```

For the threshold variable *minMotionTh* I chose to use the value three. This value depends on the configuration of the surveillance system and should be determined experimentally. Another parameter of the algorithm is the *nCorrection* offset. I used it to translate the vector magnitude interval from [-128,127] to [0, 255]. This parameter is independent of the application and I set it to 128.

Figure 61 - The illustration of optical flow of a street fight. The first row contains original frames of the fight video while the second row illustrates the optical flow of each frame.

For illustrative purposes only, I encoded the motion vectors in the HSV color space. The result of the conversion is shown in Figure 61. The hue channel encodes the motion direction and the value channel encodes the magnitude of the motion. The saturation channel was necessary to facilitate the display of the image. It is not used to store any motion information. Hence, all values in the saturation channel are set to max value. To enhance visibility, the motion vector images shown in Figure 61 are resized and interpolated.

In a similar manner, the motion vectors are encoded into an RGB image format and fetched to the input of the proposed CNN. Only the R and G channels are used for storing the motion. The B channel is not used to save motion data. All values for this channel are set to zero.

7.4 The CNN Architecture

One of the important requirements of the proposed system for detecting violence was to be suitable for embedded processing. In this regard, I analyzed several lightweight CNN architectures. That is the MobileNet [108] the SquezeNet [109] and the CNN network proposed by Krizhevsky [111].

In addition to these architectures, in the early stages of my research, I also analyzed a more complex architecture such as the one defined in paper [103]. It is designed to recognize several human actions. The solution works well both on captured videos with static surveillance cameras and those captured with dynamic PTZ (pan tilt zoom) cameras. But this approach requires huge datasets and performance hardware to build and use the model. Indeed, complex CNN are capable of recognizing more actions, but require much more training data.

Nevertheless, the MobileNet and the SqueezeNet are better fit for my application but require a fairly large input image size. Also, these networks have a large number of parameters, which involves using a large datasets to train them. Basic version of MobileNet uses 0.47 million parameters, while the compressed SqueezeNet uses 0.42 million parameters. These requirements are incompatible with my requirements. I needed an architecture that accepts small images and can be trained using a small data set.

As the forementioned networks were not suitable for my application I investigated the CNN proposed by Krizhevsky. Therefore, inspired by the work [110] that was designed to classify the CIFAR dataseet I designed a novel architecture shown in Figure 62. This architecture is a slight modification of the CNN proposed by Krizhevsky in [111]. I designed it to meet the requirements of the violence detection system. That is, to accept 16x16x3 image at the input and use a small number of parameters. The architecture has only 0.21 million parameters of which 20 418 are variables that represent weights and biases. This small number of parameters allows the network to be trained using a small number of training samples and run smoothly on an embedded architecture, such as a smart surveillance camera.

The network is feed with the MPEG flow motion vectors stored in a 16x16x3 matrix. For each 256x256 frame in a video sequence, the video encoder provides a 16x16 array representing the motion vectors. The reason for such a small array of motion vectors is due to the coding technique involved in the process. The decoder divides the frame into 16x16 blocks and generates just one motion vector for each block. Even though only two matrix channels are used for inference, I used a matrix of depth three to store motion vectors. This is only due to the fact that it was easier for me to store the motion vectors in RGB image format rather than in a two depth matrix.

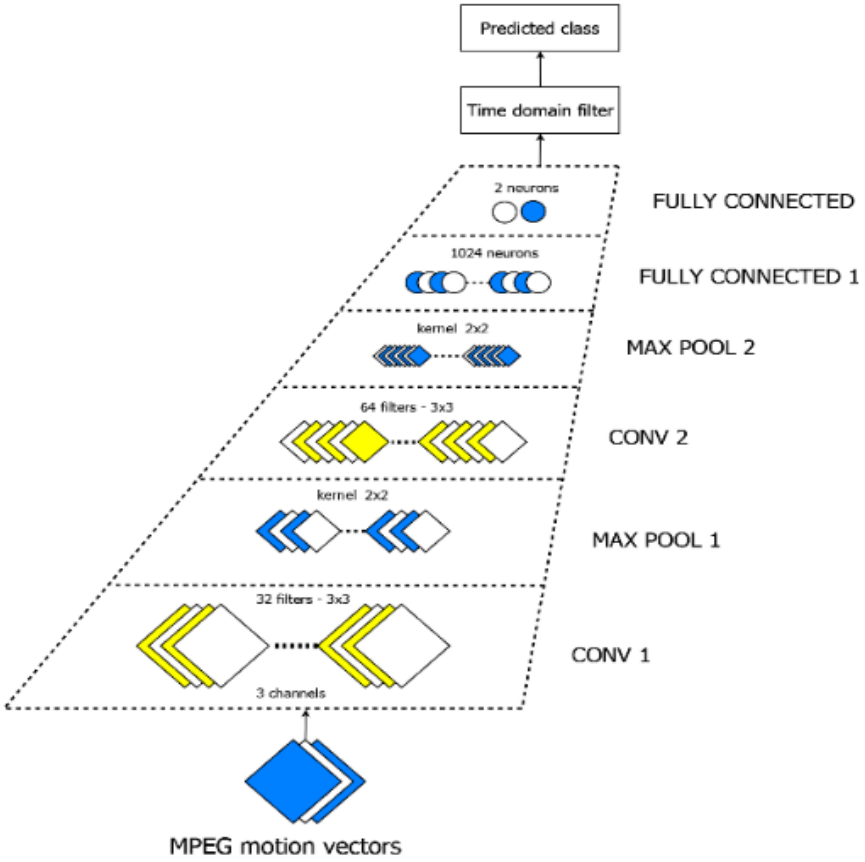


Figure 62 - Proposed CNN architecture

The convolution kernels of all layers have the size 3x3x3. The first convolutional layer of the network consists of 32 filters and the second one is composed of 64. Doubling the number of filters in the second layer leads to the extension in the channel depth. Moreover, the Rectified Linear Unit (ReLU) and Max Pooling layer follow each convolutional layer. The Max Pooling with a kernel set to 2x2 slightly reduces the size of network activation maps. Consequently, the first fully connected layer receives 4x4x64 activation maps. Finally, the last network layer is also one fully connected, but it has only two outputs. One output corresponds to fight and the other to no fight class.

7.5 Time Domain Filter

A post-processing phase consisting of two filters, linked in a cascade manner, is also part of the proposed solution for detecting violence. Data processed by the CNN is

forwarded to the input of the filter. The purpose of the filter is to capture the temporal features of the activity of the observed scene and to smooth the predictions. This allows predictions to be closer to human judgment.

The nature of a fight action directed the design of the time domain filter. The idea was to design a filter that imitates human judgments about urban fights. Commonly, the fight action is composed of short motion intervals. It contains a few unusual interactions such as punching or kicking followed by a pause with no motion. There are also particular cases of violent actions, such as wrestling, where the intensity of the motion is even more irregular. In order to declare the action belonging to the violent class, the observer must inspect the video for a while and look for patterns that contain both motion and no motion time slots.

Creating a model capable of identifying fight events in urban areas is a rather difficult task. In the solution presented here, the algorithm relies on the capabilities of the CNN to capture figting actions by using only the MPEG flow feature. However, the network output classifies some of the frames encountered in a non violent sequence, such as running and walking in a group, as violent. Moreover, prediction oscillates in the sequences that contain violent actions. To avoid such a behavior, the network output is further filtered by the means of two temporal filters. The short bursts of false predictions generated by no motion within the fight clip are eliminated by the first filter. The second one is committed to account for a minimum time required for analyzing the fight action.

Based on the assumption that the CNN output is not stable in video sequences containing fights, the following hypothesis is conceived. Measurement of the frequency of fight labels for a certain amount of time will indicate the presence of a violent action in video. The probability density estimation framework has proven to be the best choice for making the frequency measurement. Since the CNN output can not be assumed to follow a known distribution, I chose to use nonparametric density estimation. A continuous K (26) kernel estimates the density of a group that is made up of a limited number of predictions. The kernel selection criterion was optimization of measurement of the density of samples that belong to the violence class and that are enclosed by a time window. The bandwidth of the estimator is defined by the width of the time span of time window. The kernel shape is not so important. Hence, the filter uses a rectangular kernel defined by:

$$K_h(x) = \begin{cases} 1, \text{if } |x| < h \\ 0, \text{if } |x| \geq h \end{cases} \tag{26}$$

where x is the input and h is the scale parameter. The video is sampled by a time window, of size 2T+1 that has always its center situated at current time t0. Sampling is done byusing the following formula (27):

$$n(t_0;T) = \sum_t x(t)K_T(t-t_0) \tag{27}$$

The variable x(t) is the binary output provided by the CNN. It can take only two values. The value 1 stands for fight class and the value 0 represents no fight class. It is worth mentioning that the probabilities generated by the CNN can also be exploited in the action prediction process. The output of the filter is represented by the formula (28):

$$y(t\,;T,p) = \begin{cases} 1\,, \text{if } n(t;T) \geq p(2T+1) \\ 0\,, otherwise \end{cases} \tag{28}$$

The parameter *p* in the equation represents the percentage of fight predictions generated by the CNN in a 2T+1 length time window.

To adjust the first filter, it was necessary to discover the values for *p* and *T* that provide the best performance. The optimization of the second filter follows the same procedure. However, because the roles of the filters differ, the optimization criterion for the second one is not exactly the same. More details are provided in chapter 6.7.3, which is concerned with optimizing time domain filters.

7.6 The datasets

For testing the proposed approach I investigated four datasets and ended up using only two of them. The first data set that seemed appropriate for the validation of the proposed algorithm is the UCF101 data set defined in article [112]. It contains 101 human actions classes. This large number of classes led me to believe that the data set also contains human fighting actions, but I was wrong. Thus, I could not use it in my experiments. For the same reason I could not use the second data set in my experiments, namely the UCSD data set defined in article [113].

The next two data sets I investigated are the BEHAVE data set [105] and the ARENA data set [106]. These data sets are best suited for my experiments. The BEHAVE dataset was made by the computer vision students of the University of Edinburgh. They made this data set to offer the computer vision researches a common way to test the performance of their computer vision algorithms. In this regard, the video within this dataset contain ten types of human behaviors:

- **in group** – two or more people are in a group and they are not moving very much
- **approach** – the group of people approaching another group of people
- **walk together** – two or more people walking toghether
- **split** – two or more people walk toghether and then split into several smaller groups
- **ignore** – people who ignore each other

- **following** – people who follow each other
- **chase** – two or more people that form a group are chasing another group
- **fight** – two or more groups of people fighting
- **run together** – the group of people is runing together
- **and meet** – two or more people met each other

All those scenarios were acted out by the students. In the Figure 63 is an illustration of several such scenarios contained in this dataset. I believe the surveillance camera was installed on the second or third floor of the university building, with the intention of capturing the scene in a similar way to the city's surveillance cameras.

The dataset also provides ground truth bounding boxes and a markup file that contains labels of all the frames in the video. This information is very useful because the users of the data set do not need to spend time for generating the video ground truth. In terms of the number of videos, this dataset contains only one video that lasts 57 minutes. Its resolution is 640x480 pixels and the frame rate is 25 fps.

a) b) c) d)

Figure 63 - Samples from the BEHAVE dataset; a) in group scenario; b) walk together scenario; c) fighting scenario; d) running scenario

The datasets

The second dataset I used to evaluate the proposed approach is the ARENA dataset [106]. It is intended to test video analysis algorithms that aim to capture abnormal behaviors in a truck parking lot. In this sense, the authors of this dataset installed a video camera in every corner of the truck. The rectangle in the figure 64 illustrates the truck and the four light blue semicircles illustrate the field of view of the four non-overlapping on-board video cameras. Moreover, the authors used an additional surveillance video camera to capture the entire parking lot.

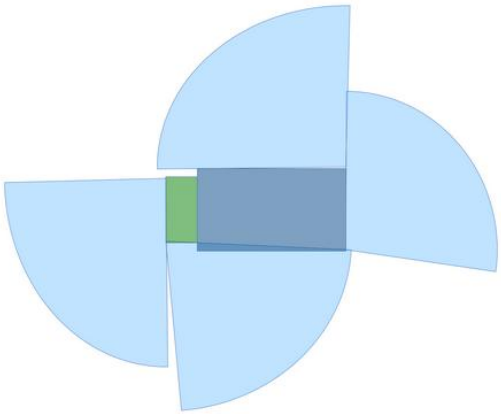


Figure 64 - The truck and the four surveillance cameras used to generate the ARENA data set

In figure 65 are some samples of the ARENA dataset. In the middle is the frame that is captured by the video camera that views the entire parking lot. In the corners of the same figure are frames captured by the four on-board cameras.

Figure 65 - Samples from the ARENA dataset.

It is worth mentioning that all the scenes in the ARENA dataset are directed. These contain the following human behaviors:

- person falling to ground
- loitering in the vicinity of the vehicle
- walking around the vehicle
- attack to driver
- stealing from vehicle

All surveillance cameras used to build this dataset have the same configuration. That is, their resolution is set to 1280 x 960 pixels, while the frame rate is set to 30 fps.

7.7 Experiments and results

7.7.1 Data Preparation

Because real threats are rare, it has been challenging for me to find datasets that contain violent actions. Moreover, the requirement that all videos within the dataset must be recorded by static surveillance cameras made this task even more difficult. However, after spending a lot of time looking for the right datasets, I came across the BEHAVE [105], ARENA [106] and UCSD [113] datasets. Although of low complexity, these data sets are suitable for training and evaluating the video analysis algorithm I propose in this chapter. Both the BEHAVE and the ARENA datasets contain violent and non-violent human actions. The UCSD dataset is slightly different. It contains unusual actions and a lot of non-violent sequences.

In order to keep the dataset in balance, I decided to exclude the UCSD from experiments since the BEHAVE and ARENA already contain sufficient non-violent scenes. Specifically, for testing the algorithm I used the BEHAVE and the ARENA datasets. While for training, I only used the BEHAVE data set. The ARENA contains only a few violent actions and is not suitable for this purpose.

To keep the balance, I grouped the sequences from the BEHAVE dataset into 22 sub-clips totaling 11872 frames. The reorganized dataset contains the following activities:

- *Attack* (seven clips)
- *Group* (four clips)
- *Run* (six clips)
- *Walk* (five clips)

I grouped all these subclips into just two classes. The no fight class is made up of *Group, Run* and *Walk* video clips whereas the fight class includes only the *Attack* video clips from the database. The violent actions are barely present in the ARENA dataset. There are just two small fighting sequences in this dataset. Each lasts only

several seconds. Therefore, I choose to use the ARENA dataset exclusively for testing.

To address the issue of the small number of videos that contain fight action, I decided to take advantage of the augmentation technique. In this way, the volume of violent clips in the BEHAVE dataset is considerably increased. The reason I chose the augmentation is that it curtails the classification model capacity thus reducing the necessity for regularization. It also supports the network to learn to be invariant to translation.

In this regard, I designed an augmentation algorithm that relies on the sliding box technique. The algorithm uses a list of region of interest (ROI) of size 256x256 to create new short videos. These regions must be defined manually so that each region in the list delimits the actors of an action at a given time in the video. In addition to the list of ROI's the algorithm also uses a list of integers. They specify the number of frames to be extracted for each region.

Figure 66 - Augmentation technique

Let's suppose that the list contains only two ROI's as shown in the figure 66. To generate the first video sequence, the algorithm crop the portion indicated by the ROI Position1 from several frames of the video. The exact number of frames used in this process is taken from the list of integers. Upon completition, the result of this process is a new short video of size 256x256. The same applies for extracting the second video sequence. But in this case the algorithm uses the Position2 instead of Position1 in the extraction process.

The idea behind this approach is that if the ROI position is conveniently changed within the frame, it is possible to "grab" a fight action over a longer period of time. Each position generates a short fight clip. Thus, it is possible to generate several fight samples from a fight activity. The only requirement is to maintain the actors within the ROI.

To further improve accuracy, I applied another augmentation to the dataset. It is the horizontal frame mirroring. I flipped the images along the horizontal axis.

Using these data augmentation techniques, I increased the number of video clips from 22 to 366. This technique is very useful because it greatly increases the variety of the dataset. I enlarged equally both the fight class and the no fight class thus resulting in a balanced dataset that contains the following video clips:

- 184 clips containing the Attack action
- 48 clips containing the Group action
- 63 clips containing the Run action
- 71 clips containing the Walk action

To prepare the data for model training, I divided the classes into two equal parts. The training data set consist of 92 videos from the fight class and 91 videos from the no fight class. Similarly, the test data set consist of 92 videos from the fight class and 91 videos from the no fight class.

7.7.2 Learning the CNN Model

To evaluate the performance of the CNN classifier in detail, I trained the network with two motion descriptors. So that the first model that I build was trained with the MPEG flow and the second one was trained with the optical flow. This approach enabled me to evaluate the classification performance of the CNN architecture and also to compare the eficency of motion descriptors.

In the training stage, I followed the same steps for both models. I used the same data and the same training process. The only difference is in the algorithms for extracting motion from the video. For training the CNN with the MPEG flow I used the motion vectors provided by the video codec, while for build the optical flow model I used the motion vectors extracted by the Farneback optical flow algorithm proposed in [116]. For the MPEG stream I didn't have to resize the matrix provided by the video codec, because the CNN was designed to accept 16x16 arrays. Unfortunately for the optical flow I had to resize the motion vectors matrix to 16x16 using bilinear interpolation. This operation was necessary because the Farneback

optical flow produces an array of size 256x256 that is incompatible with the input size of the network.

Prior to training the network I had to configure the training environment. For the batch size parameter, I chose to use size 50. I consider that this value is suitable for network training, because it is neither too small nor too high. The Adam optimizer [114] learning rate was set to 0.0001, while the remaining parameters were left to the default values [110], so *beta1* = 0.9, *beta2* = 0.999, *epsilon* = 10^{-8}. Beside these parameters I chosed to use the dropout [115] with keep probability equaling 0.5. This regularization method alongside with the data augmentation is intended to reduce the network overfitting. As a loss function I used the Softmax cross entropy with logits.

For training the CNN classifier, I used the Python programming language and the Tensorflow v1.9 GPU library. The framework ran on a PC that had the following configuration: Nvidia GFORCE 1070 ti, Intel I5, 4GB RAM, 1TB HDD. This configuration enabled me to train the network fast, because I took advantage of the graphics card and the tensorflow GPU API.

During the training process I managed to save the loss and the accuracy values of both models into the tensorboard event file. The tensorboard is a powerful tool provided by the tensoflow framework. It is a web based application that allows developers to analyse the training process of their network. Nevertheless, the loss and accuracy values were saved after each training epoch. Thus I was able to visually inspect the training flow in real time using the tensorboard tool.

The event files are not automaticaly deleted when the training ends. They remain stored in the computer memory. This tensorboard feature allowed me to extract the graphs shown in figure 67, figure 68, figure 69 and figure 70. Using these graphs I also made table 10, table 11, table 12 and table 13.

Table 10 shows the loss values of the network when trained with MPEG flow. It contains the values of the loss for both train data and test data. Because it is not practical to put all the values from the graph in a table, I only extracted the loss values that corespond to epochs 20,40,60,80,100,120,140,160 and 170. These data are sufficient to analyze the training process of the network.

		Epoch										
		20	40	60	80	100	120	140	143	144	160	170
Loss	Train	0.72	0.58	0.41	0.40	0.34	0.39	0.30	0.34	0.26	0.33	0.34
	Test	0.76	0.63	0.50	0.42	0.44	0.45	0.43	0.40	0.42	0.44	0.59

Table 10 - Loss values for multiple epochs. These values are obtained during training the network with the MPEG flow.

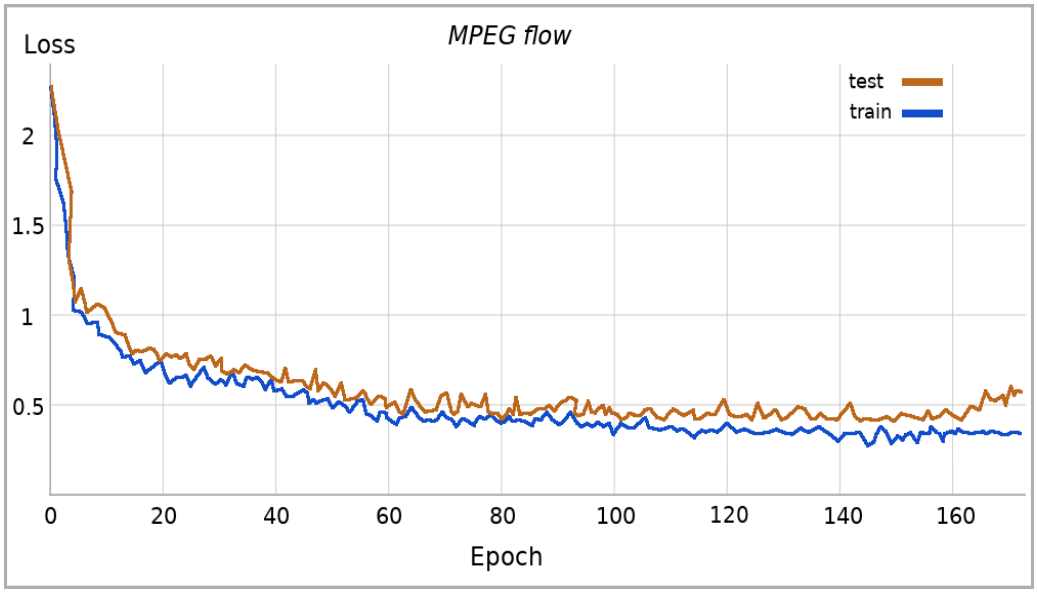


Figure 67 - Network loss obtained during training with the MPEG flow.

According to the data displayed in the table 10 and in the figure 67 by the blue line, the loss of the network during traning drops nicely until the epoch 60. After this epoch, it seems that the network is stable, which means that the loss function decreases only slightly until the end of the training, that is the epoch 173.

The drop in loss values during the training phase does not mean that the network is getting better and better. From the epoch 160 the model starts to overfit the training dataset. This phenomenon is indicated by the orange line in the figure 67 which increases starting from the epoch 160. From now on, it doesn't make sense to continue training the network. The minimum loss equals 0.26 at epoch 144 for the training dataset while for the test dataset it equals 0.40 at epoch 143.

The accuracy of the model is closely related to the loss of the model. The graph in the figure 68, as well as the data in table 11, show that the accuracy for the train dataset continues to improve as the number of epochs increases. The evolution of the accuracy for the test dataset is different from the accuracy of the training dataset.

Because of the overfitting issue the accuracy for the test dataset starts to decline at epoch 160. The maximum accuracy atained by the model during training is 82% at epochs 59 and 136 while for the testing dataset the maximum accuracy is 81% at epoch 136.

		Epoch										
		20	40	59	60	80	100	120	136	140	160	170
Accuracy	Train	0.64	0.72	0.82	0.79	0.71	0.78	0.76	0.82	0.76	0.74	0.75
	Test	0.57	0.65	0.80	0.73	0.68	0.79	0.74	0.81	0.70	0.69	0.61

Table 11 - Accuracy values for multiple epochs. These values are obtained during training the network with the MPEG flow.

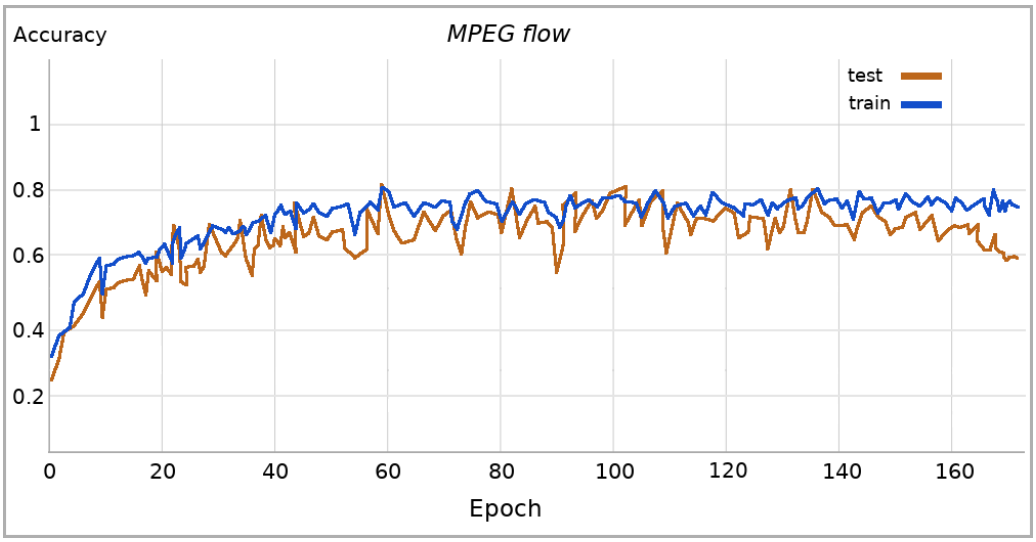


Figure 68 - Network accuracy obtained during training with the MPEG flow.

Regarding the second model, the one built using optical flow, the evolution of the loss is very similar to the evolution of the loss of the first model. In table 12 are the values of this function for epoch 20,40,60,80,100,120,140,160 and 180 for both train and test data sets.

			Epoch									
		20	40	60	80	100	120	129	140	156	160	180
Loss	Train	1.19	1.02	0.89	0.65	0.64	0.51	0.42	0.50	0.52	0.52	0.54
	Test	1.34	1.15	1.00	0.83	0.74	0.52	0.72	0.62	0.52	0.65	0.82

Table 12 - Loss values for multiple epochs. These values are obtained during training the network with the optical flow.

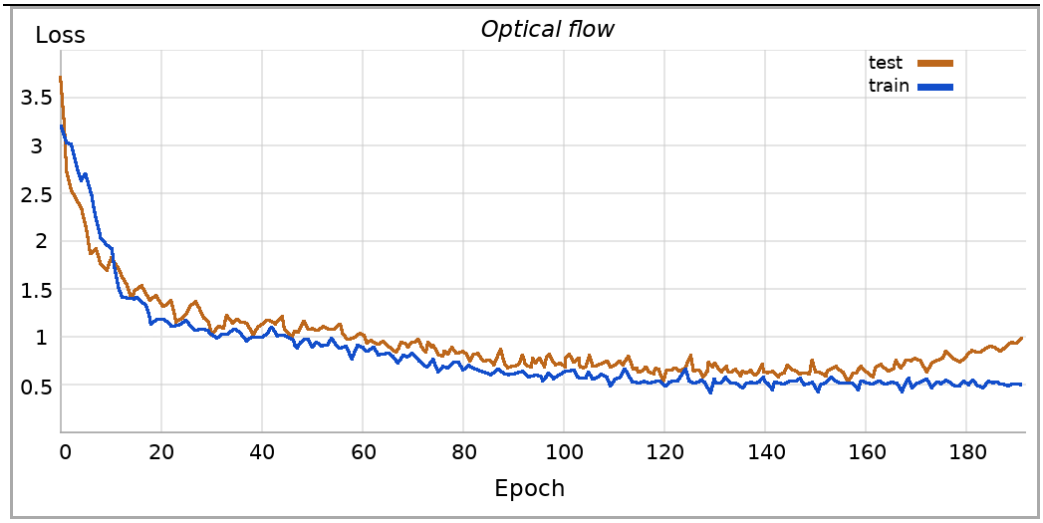


Figure 69 - Network loss obtained during training with the optical flow.

As illustrated in the figure 69 the loss decreases smoothly during training until epoch 60 for both the train data set and the test data set. After this epoch the loss for the train data set drops just for very small amount at each epoch until the end of the training, that is until the epoch 184. The loss for the test data set also decreases slowlly until the epoch 158. But, from this epoch and until the epoch 184 the loss increases.

This means that the network starts to overfit the training data set at epoch 158. Therefore, from this point on, it does not make sense to further train the network. The minimum loss equals 0.42 at epoch 129 for the training dataset while for the test dataset it equals 0.52 at epochs 120 and 156.

		Epoch										
		20	40	60	80	100	120	134	140	160	175	180
Accuracy	Train	0.54	0.60	0.68	0.65	0.64	0.69	0.67	0.66	0.68	0.72	0.65
	Test	0.47	0.52	0.56	0.58	0.64	0.61	0.69	0.60	0.64	0.56	0.55

Table 13 - Accuracy values for multiple epochs. These values are obtained during training the network with the optical flow.

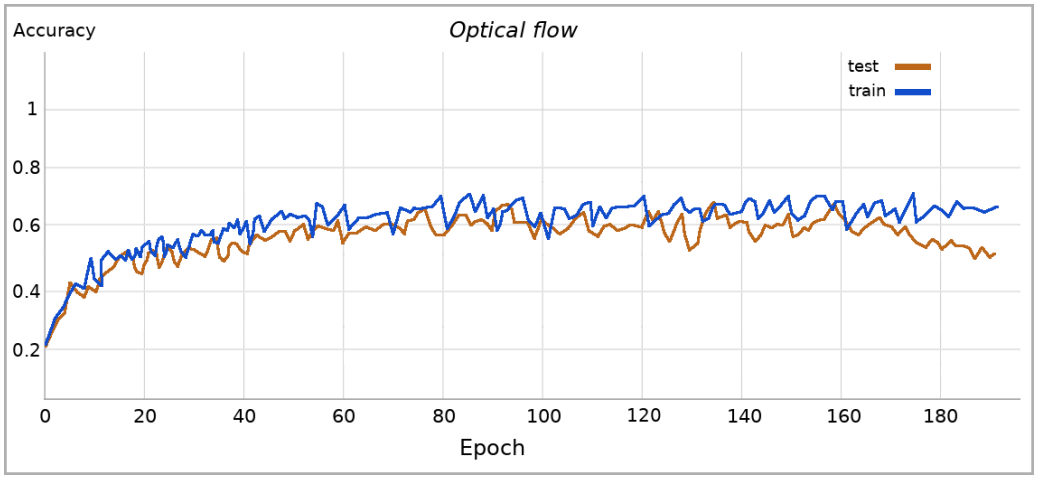


Figure 70 - Network accuracy obtained during training with the optical flow.

The accuracy of the optical flow model is displayed in the graph of figure 70. The accuracy for the train data set increases continuously until the end of training. In a similar manner the accuracy for the test data set increases but only to the epoch 158. From this point until the end of training, the accuracy of the test dataset decreases. The maximum accuracy atained by the model during training is 72% at epoch 175 while for the testing dataset the maximum accuracy is 69% at epoch 134.

These analyses of the training process of the networks reveal that the training should be stopped at epoch 143 for the MPEG flow model and at the epoch 156 for the optical flow model. To be able to use the models from these epochs I used the tensorflow saving feature. That is, I instructed the tensorflow to save the weights of the model after each epoch, so that I could use any of the generated models. Therefore, for the MPEG flow classifier I chose the model generated at epoch 143 and for the optical flow classifier I chose the model generated at epoch 156.

7.7.3 Time Domain Filter Optimization

After establishing the models of the two classifiers I proceeded with the adjustment of the time domain filter. Remember that, the time dimain filter consists of two identical filters that are aimed to smooth the output of the CNN. Because of the nature of the fight action the network output is not stable when it processes the fight videos. It oscillates between fight label and no fight label. This is normal because the fight consists of several kicks followed by pause moments in which nothing happens. The filters are designed to attenuate the oscillating output of the network thus making the predictions of the algorithm more natural.

To function properly, the first filter requires the definition of the parameters T and *p*. The T parameter stores the duration of pre and post frame inspection intervals. It is used by the filter to establish the time frame window length which is 2T+1. This parameter is closely related to the nature of the fight. Hence in order to establish the value for it I analyzed the fight videos within the dataset.

I found that a second is fair enough for a human observer to conclude that there is a fighting action in the video stream. This means that the filter will work correctly if the time frame window is equal to one second. To get one second for the time window, I chose to set the T parameter to 10. Indeed, if the time frame window length is increased, the false alarm rate decreases. But that has a cost. The video streams that contain short fights are omitted by the system, so the detection rate decreases.

The *p* is the percentile parameter that is used by the filter to differentiate the no fight sequences from the fight sequences. To filter the prediction for frame F, the filter uses the prediction of twenty one frames when T=10. That is, the predictions of the ten frames before frame F, the prediction for frame F, and the predictions of the ten frames after frame F. If in this time frame window the percentage of frames that are labeled by the network as fight exceeds the value *p* the filter labels the frame F as fight. In this way the output of the CNN is smoothed out.

The trickiest part was to choose the value for the *p* parameter. If it is too small, the filter increases the false alarm rates, while if it is too large, the filter omits short fights. To get the best value for this parameter I used the F1 (29) performance metric. It shows how accurate the model is by using precision and recall. The F1 score is calculated by the following formula:

$$F1 = 2\,\frac{precision * recall}{precision + recall} \tag{29}$$

Precision (30) is concerned about the network predictions related to the positive class that is the attack class. It shows how many fight predictions are correct out of the total fight predictions. The formula used for computing the precision is:

$$precision = \frac{TP}{TP + FP} \tag{30}$$

Wherein the *TP* corresponds to true positives (frames correctly labeled as fights) and *FP* represents false positives (frames that do not contain fight but are labeled as fight).

Recall is a metric related to the ground truth. It measures how many fight predictions the network correctly predicted from the total fight frames in the video.

$$recall = \frac{TP}{TP + FN} \tag{31}$$

Wherein the *TP* corresponds to true positives (frames correctly labeled as fights) and *FN* represents false negatives (frames that contain fight but are labeled as no fight).

The best value for the parameter *p* is the point where the F1 score reaches the maximum value. To find out this value it was necessary to calculate the precision and recall of the algorithm for several values of the parameter *p*. In this process I used only the test data set. For each *p* I made predictions of all the frames within the test dataset using the MPEG flow model and the first time domain filter. At first I stored the results provided at the output of the filter into an excel spreadsheet. Then I calculated the precision and recall for each value of the *p* parameter.

Moreover, in order to be able to compare the performance of the MPEG flow motion descriptor with the optical flow descriptor I repeated the experiment. But this time, I replaced the MPEG flow model with the optical flow model. So, the predictions of all frames within the test dataset were made using the optical flow model and the first time domain filter. The data from these experiments are displayed in table 14 and table 15. For the sake of simplicity, the pair MPEG flow model and the first filter are refered in the continuation of this manuscript as MPEG flow model. The same applies foor the optical flow model and the first filter. This pair is refered as optical flow model.

For convenience, in my experiments I used the number of frames instead of the percentage. This is why in table 14, table 15, table 16, figure 71, figure 72 and figure 73 appears the threshold. The threshold θ is nothing but *p* expressed in number of frames. The relation between the threshold and the parameter *p* is defined by equation 32.

$$\theta = p * (2T + 1) \tag{32}$$

The *p* is always positive as is the threshold. Because the threshold denotes the number of frames, it is always an integer number. In my experiments, I chose to use a set of numbers for the threshold that starts with the value 1 and ends with the value 21. In this set each number is computed by adding one to the previous number. Thus, I managed to cover the entire range of possible values for the threshold variable. I chose to use 1 as the minimum value, because this is the minimum number of frames for which the filter works correctly. The maximum value is dictated by the length of the time window which is 2T + 1 that is 21.

Threshold	Precision	
	Optical flow	MPEG flow
1	0.20	0.30
2	0.25	0.36
3	0.29	0.41
4	0.32	0.44
5	0.34	0.47
6	0.37	0.48
7	0.39	0.48
8	0.40	0.49
9	0.41	0.50
10	0.43	0.51
11	0.44	0.53
12	0.45	0.55
13	0.47	0.56
14	0.48	0.58
15	0.47	0.60
16	0.47	0.60
17	0.47	0.61
18	0.47	0.62
19	0.48	0.67
20	0.50	0.78
21	0.42	0.85

Table 14 – Precision for both the optical flow model and the MPEG flow model

Using the data from table 14 I generated the graph in figure 71 which shows the precision as a function of the threshold for both models. The precision for the MPEG flow model is represented by solid curves, whereas the precision for the optical flow model is expressed by dashed-dotted curves. I intentionally plotted the curves of both models on the same graph in order to compare them visually.

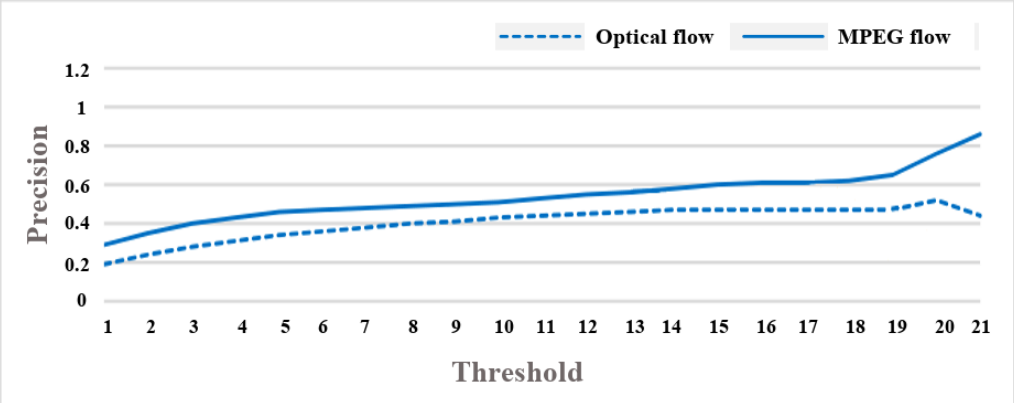


Figure 71 – Precision graph for both the optical flow model and the MPEG flow model

Up to the threshold value of twenty, for both models, the precision increases as the the threshold increases. The precision for the MPEG flow model grows even more rapidly starting from the threshold value of nineteen.

This is a clear indication that for the MPEG flow model, higher values of this parameter make fight predictions more precise. The number of true positives increases and the number of false positives decreases. Which means that false alarm rates are going down. This rule also applies to optical flow model, but only up to the threshold value of twenty. For threshold values greater than twenty, the precision for the optical flow model begins to decrease.

Threshold	Recall	
	Optical flow	MPEG flow
1	0.96	1.00
2	0.92	1.00
3	0.91	1.00
4	0.90	1.00
5	0.89	1.00
6	0.88	1.00
7	0.83	1.00
8	0.78	0.98
9	0.74	0.97
10	0.71	0.94
11	0.68	0.92
12	0.63	0.90
13	0.60	0.87
14	0.54	0.83
15	0.49	0.79
16	0.44	0.75
17	0.37	0.63
18	0.28	0.54
19	0.22	0.45
20	0.17	0.36
21	0.08	0.20

Table 15 – Recall for both the optical flow model and the MPEG flow model

For the recall, I generated the graph in figure 72 using the data from the table 15. The recall for the MPEG flow model is represented by solid curves, whereas the recall for the optical flow model is expressed by dashed-dotted curves. The values for the recall are high for both models when the threshold is low. For threshold values up to eight, the recall for the MPEG flow model is constant and is equal to one. From this point on, the increase in the threshold value decreases the recall value of the MPEG flow model.

For the optical flow model, the recall decreases continuously. Up to the threshold value of six, the recall decreases only by a small amount. From this point on, the recall decreases more abruptly for each subsequent threshold.

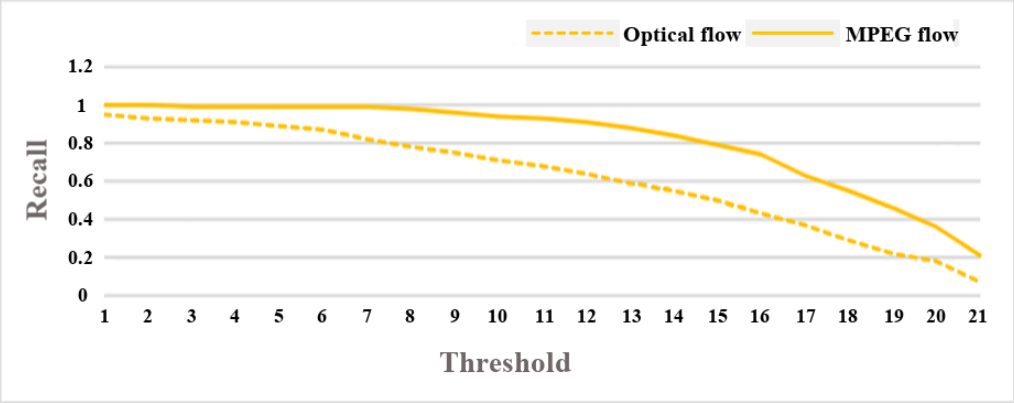


Figure 72 – Recall graph for both the optical flow model and the MPEG flow model

Using the values for precision and recall from tables 14 and 15 I computed the values for the F1 score. These are shown in table 16.

Threshold	F1 score	
	Optical flow	MPEG flow
1	0.33	0.46
2	0.39	0.53
3	0.44	0.58
4	0.47	0.61
5	0.49	0.64
6	0.52	0.65
7	0.53	0.65
8	0.53	0.65
9	0.53	0.66
10	0.54	0.66
11	0.53	0.67
12	0.53	0.68
13	0.53	0.68
14	0.51	0.68
15	0.48	0.68
16	0.45	0.67
17	0.41	0.62
18	0.35	0.58
19	0.30	0.54
20	0.25	0.49
21	0.13	0.32

Table 16 – F1 score for both the optical flow model and the MPEG flow model

F1 scores for both the MPEG flow model and the optical flow model evolve similarly. They slowly increase in value and then at some point begin to fall sharply. The graph

in figure 73 shows the evolution of these scores. The F1 score for the MPEG stream model increases to the point when the threshold value is twelve. For treshold values between twelve and fourteen, the value of the score remains constant. After this point that is for threshold values greater than fourteen, the F1 score for the MPEG flow model begin to fall sharply.

A similar pattern follows the score for the optical flow model. It increases slowly until the threshold value reaches nine. The score stabilizes for a while and then starts to decrease for threshold values greather than eleven.

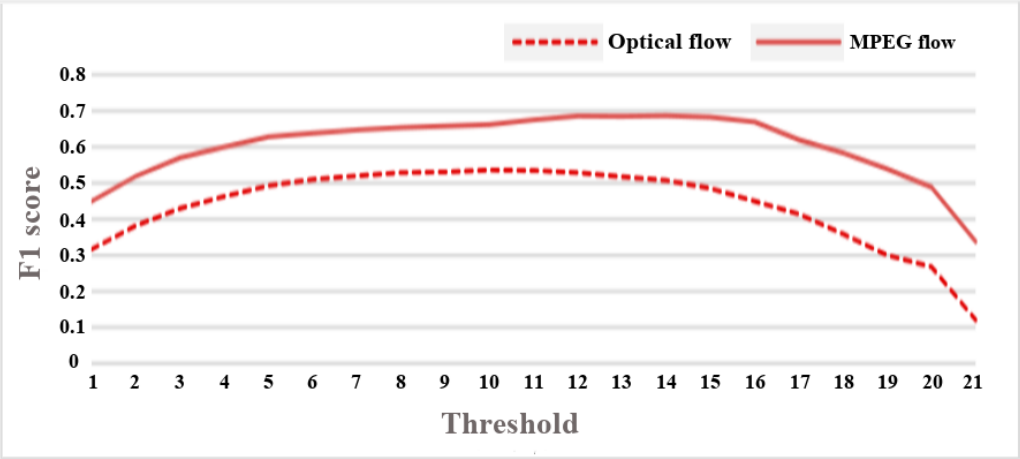


Figure 73 – F1 score graph for both the optical flow model and the MPEG flow model

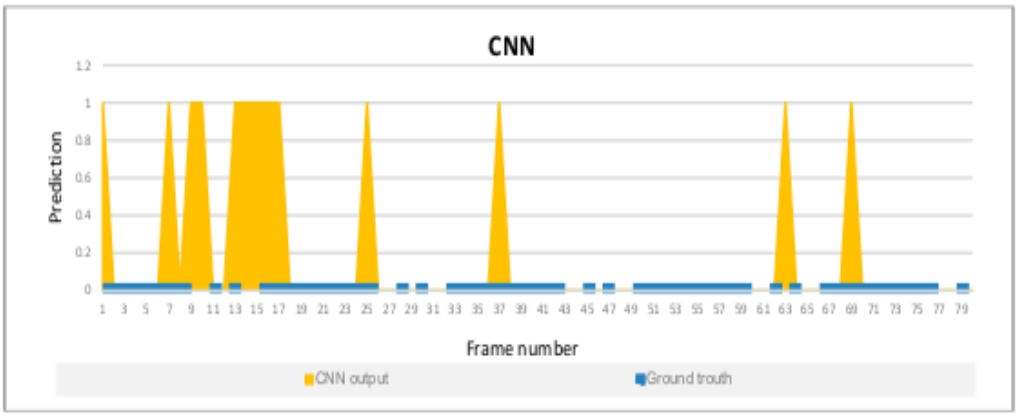
Based on these data, I conclude that for the MPEG flow model the best value for the threshold θ of the first time domain filter is 14. For this value, the F1 score reaches the maximum value as shown in graph of figure 73. In this setting the parameter *p* becomes equal to 66%.

It can be easily noticed that, regardless of threshold value, precision, recall and consequently the F1 score have higher values for MPEG than for Farneback optical flow. This is a clear indicator that the MPEG flow motion descriptor outperforms the optical flow descriptor. For this reason, I was no longer interested in the optical flow model, so I did not use it any more in my subsequent experiments.

Hence for adjusting the second time domain filter I used only the MPEG flow model and the first time domain filter. The second time domain filter is designed to filter the noise. In some no fight sequences, the noise generated by short motions is still present in the output of the first filter.

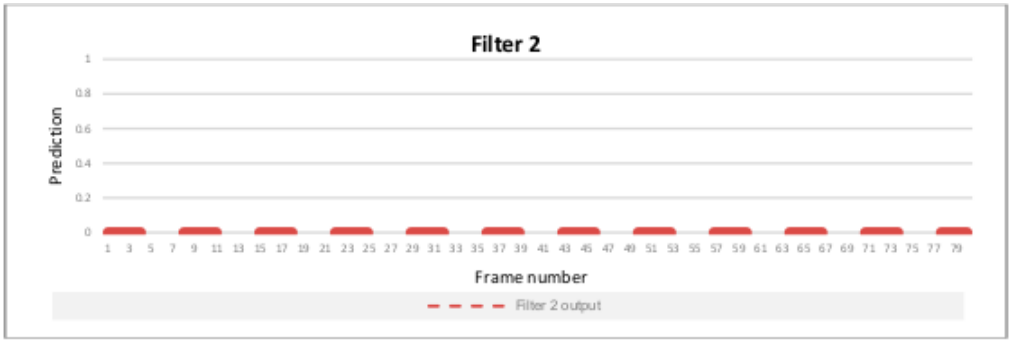
One such case is illustrated in figure 74. The graph in figure 74.a displays the output of the CNN network. You can notice that it contains several spikes. Most are at the beginning of the chart. All these spikes are generated by the noise that is present in the video frame. In some frames, noise is perceived by the MPEG flow as motion that is similar to fight motion. These frames are labeled by the CNN as belonging to the fight class which is incorrect.

The first filter manages to filter right many of the false predictions generated by noise, such as predictions for frames 13, 14, 15, 16, 17, 25, 37, 63 and 69. But in cases where false predictions are more frequent, like for frames 1, 7, 9, and 10, the filter fails to filter them correctly. This behavior of the first filter can be observed in the graph of figure 74.b. The graph illustrates the output of the second filter.



a)

b)



c)

Figure 74 - Output of the algorithm stages; a) CNN output; b) Filter 1 output; c) Filter 2 output.

To remedy this problem, I added a second time domain filter to the algorithm. This filter eliminates parasitic impulses using a much longer time interval for the analysis. Its operation is identical to the operation of the first filter. The filter uses a time window and a percentile parameter *p*. It compares the percentage of fight predictions found in the time window with the percentile parameter. Only when the percentage equals the percentile *p* the filter outputs a fight label.

I have optimized the filter parameters so as to minimize the false positive rate and keep the recall at 100%. Notice that the optimization is done by implying the labeling of predictions at video clip level. I chose to use 100% for the percentile parameter and for the time window I chose length seven. In this configuration, to filter the prediction for frame F, the filter uses the prediction of seven frames. That is, the predictions of the three frames before frame F, the prediction for frame F, and the predictions of the three frames after frame F.

This filtering technique allowed me to eliminate false predictions generated by noise, which were filtered incorrectly by the first filter. In the graph of figure 74.c are the outputs of the second filter. You may notice that the false predictions are gone.

7.7.4 Inference on Raspbery PI

After training the model and setting the values for the filters, the algorithm is ready for the testing phase. In this phase I used the following hardware: a Raspberry PI 3 with a 1.2 GHz Quad Core processor and 1 GB of RAM, a USB video camera and, of course, a moitor, a keyboard and a mouse.

Because the algorithm is meant to run on hardware with limited resources, I wanted to test it on such a device and not on a PC. Since I had a Raspberry PI 3 at hand, which is a mini computer with limited resources, I decided to use it to test the algorithm.

For the software part I used the QtCreator 4.2.0 integrated development environment, the OpenCV 3.4.0 computer vision library, the gcc 6.2.1 compiler and the C++ programming language. Implementing the algorithm in this development environment was an easy task. To load the MPEG flow model that I generated with the tensoflow API I used the dnn module of the OpenCV library. I also used this library for basic image processing tasks like loading videos, displaying frames or loading video stream of the USB camera. The two time domain filters are implemented using buffers and arithmetic operations that are provided by standard C++ programming language libraries.

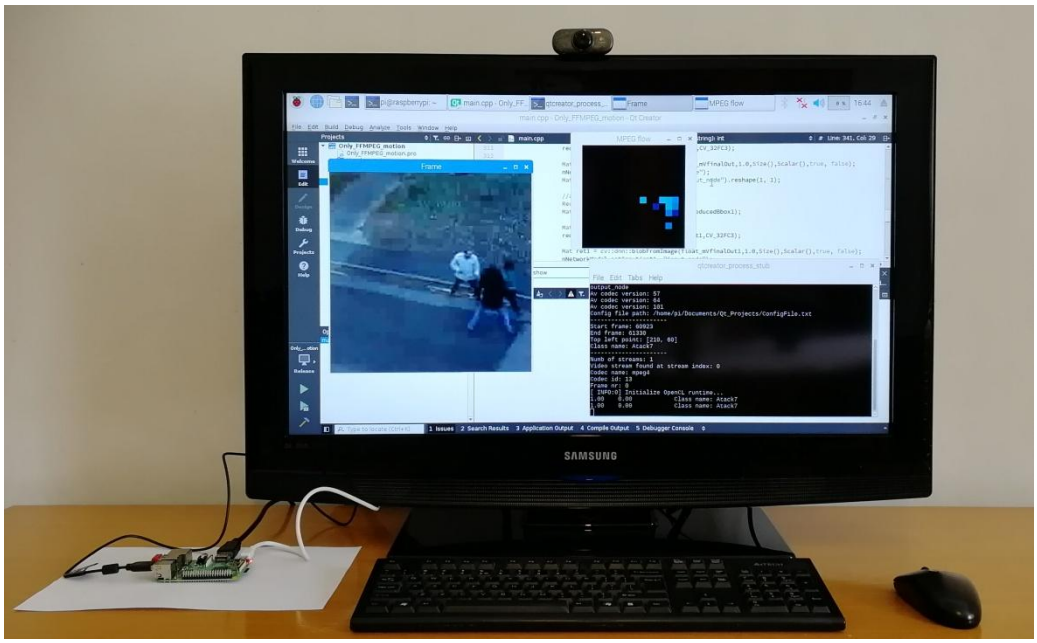


Figure 75 – The hardware used to test the algorithm

To test the algorithm in detail, I performed two types of tests. Both are based on the predictions at video clip level and use the MPEG flow model and the two time domain filters. The first type of test uses videos from the datasets and the second type of test uses live streams captured by the USB camera.

For the first type of test, I used some of the videos from the BEHAVE dataset [105] and the ARENA dataset [106]. That is, from the BEHAVE dataset I used only the test data set, because it did not make sense to test the algorithm with the train data set. The BEHAVE dataset was already familiar to me. Therefore, it did not take much work to extract the relevant data from it. I already divided it into the train data set

and the test data set when I trained the network. All I had to do was to extract the testing data set and divide it into video clips.

I divided the test dataset into clips because I wanted to compare the detection performance of the algorithm with the performance of similar algorithms from the literature, such as those defined in papers [117] and [118]. The authors of these papers report the performance of their algorithm at video clip level. So I needed to use the same testing approach.

The analysis of the test data set resulted in 86 videos. In order to extract these videos I used the criterion defined in article [117]. Therefore, each clip includes one of the following types of activities: run, group, walk and fight. Obviously, the fight clips belong to the fight class, while the no fight class contains the run, group and walk activities. The algorithm labels a clip as fight only if at least one fight label is present in the output of the second filter. It is important to mention that I have excluded frames that do not contain any motion from the videos. Their classification is insignificant and rather trivial for the purpose of this work.

		Labeled class	
		Fight	*No fight*
Predicted class	Fight	15	19
	No fight	0	52

Table 17 – Confusion matrix of the proposed algorithm [BM4]

After preparing the data set, I proceeded with testing the algorithm. The results for the 86 videos are shown in Table 17. These are quite satisfactory. Recall attaied 100%, the false positive rate is only 26.76% and the acuracy is 77.9%. This means that all fight events are captured by the algorithm so it doesen't miss any of them. Regarding false alarms, the score is 26.76%, which is not bad. Only a small number of video clips were incorrectly predicted. Instead of being classified as belonging to the no fight class, the algorithm classified them as belonging to the fight class.

For a fair comparison with other works reported in literature, I did another set of experiments. This time I wanted to adjust the algorithm so as to obtain the best accuracy. To do so, I only changed the threshold value of the first filter. Instead of using the value 14, I used the value 21. The threshold value of the second filter remains set to 7 as for the first experiment. In this configuration, the algorithm obtained an accuracy of 86.93% for the 86 videos from the BEHAVE dataset. This result is excelent.

To evaluate the performance of the algorithm, I compared the obtained results with the results of similar methods found in the literature. It was very difficult for me to

find computer vision algorithms that use the BEHAVE dataset. But after a thorough research I managed to find a few. The performance of these algorithms as well as the performance of my algorithm are shown in table 18.

Algorithm		*ACC±SD*	*AUC*
Existing algorithms	HOG+BoW [53]	58.69±0.35%	0.6322
	HOF+BoW [53]	59.91±0.28%	0.5893
	HNF+BoW [53]	57.97±0.31%	0.6089
	ViF [53]	82.02±0.19%	0.8592
	MoSIFT+BoW [112]	62.02±0.23%	0.6578
	RVD [163]	85.29±0.16%	0.8878
	AMDN [158]	84.22±0.17%	0.8562
	MoWLD+BoW [117]	83.19±0.18%	0.8517
	MoWLD+SparseCoding [117]	85.75±0.15%	0.8891
	MoWLD+KDE+SparseCoding [117]	87.17±0.13%	0.8993
	Proposed method	86.93±0.21%	0.9543

Table 18 - My solution compared to similar approaches in the literature using the BEHAVE dataset.

Zhang's [117] paper states that it attains the 87.17% accuracy on the BEHAVE dataset but it doesent provide the recall. This metric is much more significant than the accuracy for a violence detection system. Another solution [118] using the same dataset provides results through the graph. It plots true positive rate as a function of false positive rate. The graph shows that for true positive rate of 100%, the recall is almost identical to that obtained by the algorithm proposed in this thesis.

All these findings convinced me that the algorithm I proposed in this chapter works excellently on the BEHAVE dataset. But I wanted to test it on another data set, to make sure that the algorithm has good generalization performance. For this purpose I used the ARENA dataset [106]. This data set was not the most suitable for this task, but it was all I had. It contains only two short fight actions, which last only a few seconds. The algorithm succesfully recognized both actions. It also correctly predicted ten videos from this dataset that did not contain fights. These are excelent results. The recall as well as the accuracy of the algorithm is 100% for this dataset.

At this point I ran out of test data and I wasn't very convinced that the algorithm generalizes well. Since I had no other options, I decided to test the algorithm on my own data. This is the the second type of test that uses video stream captured by the USB camera.

Experiments and results

For a successful test, I had to set up the camera first. The algorithm is designed to process videos that are encoded by the MPEG-4 video codec. Hence, in order to work properly I had to configure the camera to encode the video in this format. This was an easy task. I just had to select the MPEG-4 codec in the camera software settings. Also through these settings I configured the camera resolution to 640 x 480 pixels at 25 fps. This setting provides the best results because the MPEG stream model used in the algorithm was trained at this resolution.

To test the algorithm using live data captured by the camera, I installed the system in an outdoor laboratory environment. Afterwards I simulated the fight and no fight actions with a colleague of mine in front of the camera. The test was madeup of 10 scenarios. Five of them involved fights, and the rest were normal. The system captured all events correctly. It did not generate false alarms not missed any fight event. I was very pleased with these results. The recall and accuracy of the algorithm are both 100%.

The algorithm that I propose in this chapter is also very efficient in terms of processing speed. To extract the MPEG motion vectors from a frame, the ffmpeg library used by the algorithm consumes only 2 ms. The average inference time for a frame is also low, being only 19,783 ms. In terms of hardware resources, the algorithm is good in this regard as well. During the inference it uses only about 85MB of RAM and around 48% of CPU.

8 Conclusions

In this thesis I investigated diferent approaches that use compter vision in order to recognize human behavior in video. The goal was to propose a wireless video surveillance system based on a network of smart surveillance cameras to automatically recognize human behaviors in the video.

Such an approach has many benefits. The system can be easily deployed because it does not require many electrical wiring. The data transfer in the network is very low because video analysis is done by each smart surveillance camera. Moreover, the system is easely extensible in terms of functionality. Extending its functionality to recognize a new human behavior only involves loading the new computer vision algorithm on all smart surveillance cameras. From the user's perspective, the system greatly improves the eficency of the surveillance officers. They should only review the events reported by the video analysis algorithm instead of continuously inspecting the video streams.

The problem I encountered during this research was that all the algorithms I found in the literature either had low detection performance or were not designed to run on low-resource hardware, such as smart surveillance camera. Therfore, I designed new computer vision algorithms. They consume very little hardware resources and are very effective in detecting behavior in video.

The first computer vision algorithm I designed is a basic behavior classification algorithm capable of detecting and counting vehicles in traffic monitoring applications. It can count vehicles and discriminate among small and large vehicles during daytime traffic surveillance. In the detection process the algorithm extracts vehicle features using foreground extraction algorithm and contour extraction algorithm and stores them in a history buffer. It also uses a correction method in the post processing phase to increase the recall. These operations are of low complexity, so that the algorithm consumes very little hardware resources during operation. The algorithm was tested on two datasets that I made and obined excelent results.

The second computer vision algorithm I designed is more advanced than the first one, because it is based on detecting the behavior of objects in the scene. It is aimed for increasing the public security in the big cities. Specifically, the algorithm is able to automatically detect a dog attack on humans by analyzing motion in video. The solution uses object trajectories and the SVM classifier to detect the dog attack behavior.

To extract the trajectories, the algorithm employs the foreground extraction algorithm and the contour extraction algorithm. It also involves a simple silhouette based object classification phase to distiungishing human trajectories from dog trajectories. These techniques do not consume much hardware resources, thus allowing the algorithm to run on the smart surveillance camera. In addition to this benefit, the algorithm also has a high detection performance. It has been extensively tested using 119 video clips collected from public internet sources.

Experiments and results

The last computer vision algorithm I managed to design, is meant to detect complex human behavior in video. More preciselcy, the algorithm is able to automatically detect fight events in urban areas. It is hybrid approach that uses a deep neural network (DNN) and handcrafted features.

The approach defines a novel principle of using only the motion features generated by the MPEG stream to power the DNN with data. Therefore, instead of using the optical flow which is computationally expensive the solution uses the MPEG flow offered by the video decoder. Since MPEG flow is computed during video decoding, the processing power required for obtaining it, after a frame was decoded, equals to zero. Thus, the proposed approach is suitable for use in a distributed, low resource, processing system.

The architecture involves a cascade of two filters, namely a deep neural network and a time domain classifier. Spatial data processing is separate from time domain processing. The advantage offered by using only the motion features is that the illumination changes and variations of the color spectrum do not affect the performance of the algorithm. Moreover, I believe that this approach allows the solution to be easily adapted for night vision surveillance, such as infrared video surveillance.

To evaluate the proposed approach, I tested the solution using a Raspberry Pi. The experimental results showed that the algorithm attained state of the art performance. In this process I used two publicly available data sets as well as my own data that I generated using the USB camera.

Because the proposed algorithm for detecting human fight behavior is designed to work only with static cameras, any camera motion will disturb its operation. Therefore, the algorithm is not suitable for applications that use dynamic cameras. Another drawback is that it cannot detect crowd fights. This is normal because the CNN model has not been trained to recognize such events. Indeed, the datasets used in performance evaluation process contain fights that involve several people, but they do not contain crowd violence.

8.1 Contributions

- I have developed a method for detecting violent behavior in urban areas using machine learning algorithms run on a network of video sensors with limited computational resources.
- I have studied various algorithms for tracking the moving entities in a video having various speeds – implying various driving behaviors. Case study: Urban traffic surveillance application.
- I have studied detection of violent actions in urban areas, focusing on movements generated by several entities that interact with each other. Case study: Attacks conducted by stray dogs in urban environment.
- I propose a new approach to develop video surveillance using a network of smart surveillance cameras. The system is event based. Sensor node uses a video analysis algorithm to detect events in the video.
- I compared the performance obtained by using two motion descriptors as optical flow motion descriptor and MPEG motion descriptor.
- I propose an augmentation algorithm to increase the volume of the video dataset used for training the network.
- I designed an efficient algorithm to count the vehicles in video independent of driving behavior. It uses computer vision techniques to avoid counting a vehicle more than once.
- I have developed a correction method for solving the contour splitting issue. This problem is frequent in traffic surveillance and afects negatively the performance of the video analysis algorithms.
- I developed a simple but effective shape based method for classifying humans and dogs in video suitable to be run on low computational environment.
- I analyzed and compared the performance of various background subtraction algorithms in the process of separating moving entities from static background.
- I created two video databases for testing the traffic surveillance applications
    - Video footage of the intersection between Str. Stefan Cel Mare and Str. Stefan Octavian Iosif in Timisoara (41 minutes).
    - Video fooatge of the loop of the Pancevo Bridge that connects the Pancevo Bridge with Despot Stefan Boulevard in Belgrade, Serbia (43 minutes)

8.2 Publications

[BM1] Áron Virginás-Tar, Marius Baba, Vasile Gui, Dan Pescaru, Ionel Jian, "Vehicle Counting and Classification for Traffic Surveillance using Wireless Video Sensor Networks", Conference: 22nd Telecommunications Forum (TELFOR), 25-27 November 2014, SAVA Center, Belgrade, Serbia; Published in IEEE XPLORE Digital Library; Indexed in the ISI Web of Science.

[BM2] Marius Baba, Dan Pescaru, Vasile Gui, Ionel Jian, "Stray Dogs Behaviour Detection in Urban Area Video Surveillance Streams", Conference: 12th IEEE International Symposium on Electronics and Telecommunications (ISETC), 27-28 October 2016, Timisoara, Romania; Published in IEEE XPLORE Digital Library; Indexed in the ISI Web of Science.

[BM3] Marius Baba, Vasile Gui, Dan Pescaru, "Deep Learning Approach for Violence Detection in Urban Areas", Conference: 1st International Conference on Computational Methods and Applications in Engineering (ICCMAE), 23 – 26 May 2018, Timisoara, Romania; Published in ITM Web of Conferences; Indexed in the ISI Web of Science.

[BM4] Marius Baba, Vasile Gui, Cosmin Cernazanu, Dan Pescaru "A Sensor Network Approach for Urban Violence Detection Using Deep Learning", Journal: Sensors, Volume 19, Issue 7; Published by MDPI Switzerland, 8 April 2019. Journal impact factor 3.275.

[BM5] Marius Baba, PhD Research Report 1, Tracking Solution in Video-Based Sensor Networks

[BM6] Marius Baba, PhD Research Report 2, Tracking Solution in Video-Based Sensor Networks

9 Bibliography

[1] Rhodes, Susan R. „Age-related differences in work attitudes and behavior: A review and conceptual analysis", Psychological Bulletin, Vol 93(2), Mar 1983, 328-367.
[2] Naomi Ellemers,Dick de Gilder, Henriette van den Heuve "Career-Oriented Versus Team-Oriented Commitment and Behavior at Work", Journal of Applied Psychology,1998.
[3] Chaaraoui, Alexandros André, Pau Climent-Pérez, and Francisco Flórez-Revuelta. "A review on vision techniques applied to human behaviour analysis for ambient-assisted living." Expert Systems with Applications 39.12 (2012): 10873-10888.
[4] Moeslund, T. B., Hilton, A., & Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. Computer Vision and Image Understanding, 104, 90–126.
[5] D. Elliott, "Intelligent video solution: a definition", Security Magazine, 47(6), pp.46–48, June 2010.
[6] S. T. Londei, J. Rousseau, F. Ducharme, A. St-Arnaud, J. Meunier, J. Saint-Arnaud, and F. Giroux, "An intelligent videomonitoring system for fall detection at home: perceptions of elderly people", Journal of Telemedicine and Telecare, vol. 15, no. 8, pp. 383–390, 2009.
[7] Debard, Glen, et al. "Camera-based fall detection on real world data." Outdoor and large-scale real-world scene analysis. Springer, Berlin, Heidelberg, 2012. 356-375.
[8] Huang, T.; Han, Q.; Min, W.; Li, X.; Yu, Y.; Zhang, Y. Loitering Detection Based on Pedestrian Activity Area Classification. Appl. Sci. 2019, 9, 1866.
[9] Foggia, Pasquale, et al. "A Method for Detecting Long Term Left Baggage based on Heat Map." VISAPP(2).2015.

Bibliography

[10] https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html.
[11] Chaker, Rima, Zaher Al Aghbari, and Imran N. Junejo. "Social network model for crowd anomaly detection and localization." Pattern Recognition 61 (2017): 266-281
[12] Hao, Yu, et al. "Effective crowd anomaly detection through spatio-temporal texture analysis." International Journal of Automation and Computing 16.1 (2019): 27-39.
[13] http://mha.cs.umn.edu/proj_events.shtml#crowd
[14] Chan, Antoni, and Nuno Vasconcelos. "Ucsd pedestrian dataset." IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI) 30.5 (2008): 909-926.
[15] Serhan Cosar, Giuseppe Donatiello, Vania Bogorny, Carolina Garate, Luis Alvares, François Bremond. "Toward abnormal trajectory and event detection in video surveillance." *IEEE Transactions on Circuits and Systems for Video Technology* 27.3 (2016): 683-695.
[16] D. P. Chau, F. Brémond, M. Thonnat, and E. Corvée, "Robust mobile object tracking based on multiple feature similarity and trajectory filtering," in VISAPP 2011 - Proceedings of the Sixth International Conference on Computer Vision Theory and Applications, Vilamoura, Algarve, Portugal, 5-7 March, 2011, pp. 569–574.
[17] S. Zaidenberg, B. Boulay, C. Garate, D. P. Chau, E. Corvee, and F. Bremond, "Group interaction and group tracking for video surveillance in underground railway stations," in International Workshop on Behaviour Analysis and Video Understanding (ICVS 2011), Sophia Antipolis, France, Sep. 2011, p. 10.
[18] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," Science, vol. 315, pp. 972–976, 2007.
[19] Mind's Eye dataset [Online]. Available: http://www.visint.org/
[20] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust realtime unusual event detection using multiple fixed-location monitors," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 3, pp. 555–560, March 2008.
[21] J. Varadarajan, R. Emonet, and J.-M. Odobez, "A sequential topic model for mining recurrent activities from long term video logs," International Journal of Computer Vision, vol. 103, no. 1, pp. 100–126, May 2013.
[22] Zerrouki, N., Harrou, F., Sun, Y., & Houacine, A. Vision-based human action classification using adaptive boosting algorithm. *IEEE Sensors Journal*, *18*(12), 5115-5121. (2018).
[23] N. Zerrouki, F. Harrou, Y. Sun, and A. Houacine, "Accelerometer and camera-based strategy for improved human fall detection," Journal of medical systems, vol. 40, no. 12, p. 284, 2016.
[24] B. Kwolek and M. Kepski, "Human fall detection on embedded platform using depth maps and wireless accelerometer," Computer methods and programs in biomedicine, vol. 117, no. 3, pp. 489–501, 2014.
[25] E. Casilari, J. A. Santoyo-Ramón, and J. M. Cano-Garćia, "Analysis of a smartphone-based architecture with multiple mobility sensors for fall detection," PLoS one, vol. 11, no. 12, p. e0168069, 2016.
[26] Bobick, Aaron F., and James W. Davis. "The recognition of human movement using temporal templates." IEEE Transactions on pattern analysis and machine intelligence 23.3 (2001): 257-267.
[27] M. Hu, aVisual Pattern Recognition by Moment Invariants,o IRE Trans. Information Theory, vol. 8, no. 2, pp. 179-187, 1962.
[28] Gorelick, Lena, et al. "Actions as space-time shapes." IEEE transactions on pattern analysis and machine intelligence 29.12 (2007): 2247-2253.
[29] Maria Andersson, Luis Patino, Gertjan J. Burghouts – "Activity Recognition and Localization on a Truck Parking Lot", the 10th IEEE International Conference on Advanced Video and Signal Based Surveillance, 2013.

[30] Z. Zivkovic. "Improved adaptive Gaussian mixture model for background subtraction", In the proceedings of the 17th International Conference on Pattern Recognition ICPR'04, 2004.
[31] P. Dollar, S. Belongie and P. Perona, "The Fastest Pedestrian Detector in the West", BMVC, 2010.
[32] S. S. Blackman and R. Popoli, "Design and Analysis of Modern Tracking Systems", Artech House, 1999.
[33] R. Duda, P. Hart and D. Stork, "Pattern Classification and Scene Analysis", 2nd edition, Wiley, 1995.
[34] I. Laptev, "On Space-Time Interest Points, International Journal of Computer Vision", 2005.
[35] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference and Prediction, 2nd edition, Springer, 2009.
[36] Enrique Bermejo Nievas, Oscar Deniz Suarez, Gloria Bueno García, Rahul Sukthankar, "Violence Detection in Video Using Computer Vision Techniques", International Conference on Computer Analysis of Images and Patterns, 2011.
[37] Laptev, I.: On space-time interest points. In: International Journal of Computer Vision. vol. 64, pp. 107–123 (2005).
[38] Chen, M., Hauptmann, A.: MoSIFT: Recognizing human actions in surveillance videos. Tech. rep., Carnegie Mellon University, Pittsburgh, USA (2009).
[39]V. M. Arceda, K.F. Fabian, J.C. Gutierrez, "Real time violence detection in video", International Conference on Pattern Recognition Systems (ICPRS-16), 2016.
[40] B. D. Lucas and T. 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, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.
[41] B. K. Horn and B. G. Schunck, "Determining optical flow," in 1981 Technical symposium east. International Society for Optics and Photonics, 1981, pp. 319–331.
[42] C. Liu, "Beyond pixels: exploring new representations and applications for motion analysis," Ph.D. dissertation, Citeseer, 2009.
[43] Heng Wang, Alexander Kläser, Cordelia Schmid, Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. International Journal of Computer Vision, Springer Verlag, 2013, 103 (1), pp.60-79.
[44] Farnebäck G. Two-Frame Motion Estimation Based on Polynomial Expansion. In: Bigun J., Gustavsson T. (eds) Image Analysis. SCIA 2003. Lecture Notes in Computer Science, vol 2749. Springer, Berlin, Heidelberg.
[45] Shi J, Tomasi C. Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition, 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.
[46] Huangkai Cai, He Jiang, Xiaolin Huang, Jie Yang, "Violence Detection based on Spatio-Temporal Feature and Fisher Vector", Chinese Conference on Pattern Recognition and Computer Vision (PRCV) 2018.
[47] Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Dense trajectories and motion boundary descriptors for action recognition. International Journal of Computer Vision 103(1), 60–79 (2013).
[48] Kantorov, V., Laptev, I.: Efficient feature extraction, encoding, and classification for action recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 2593–2600 (2014).
[49] Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8 (2008).

[50] Nchez, J., Perronnin, F., Mensink, T., Verbeek, J.: Image classification with the fisher vector: Theory and practice. International Journal of Computer Vision 105(3), 222–245 (2013).
[51] Arthur, D., Vassilvitskii, S.: k-means++:the advantages of careful seeding. In: Eighteenth Acm-Siam Symposium on Discrete Algorithms. pp. 1027–1035 (2007).
[52] Nievas, E.B., Suarez, O.D., Garca, G.B., Sukthankar, R.: Violence detection in video using computer vision techniques. In: International Conference on Computer Analysis of Images and Patterns. pp. 332–339 (2011).
[53] Hassner, T., Itcher, Y., Kliper-Gross, O.: Violent flows: Real-time detection of violent crowd behavior. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 1–6 (2012).
[54] K. Fukushima: "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position", *Biological Cybernetics*, **36**, pp. 193-202 (April 1980).
[55] Karpathy, Andrej. 2015. "Neural Networks Part 1: Setting Up the Architecture." Notes for CS231n Convolutional Neural Networks for Visual Recognition, Stanford University. http://cs231n.github.io/neural-networks-1/.
[56] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.
[57] Yosinski, Jason, et al. "Understanding neural networks through deep visualization." *arXiv preprint arXiv:1506.06579* (2015).
[58] The cost function [Online]. Available: https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220
[59] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In Advances in Neural Information Processing Systems, pages 568–576, 2014.
[60] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deepinto convolutional nets. InProc. BMVC., 2014.
[61] Yue-Hei Ng, Joe, et al. "Beyond short snippets: Deep networks for video classification." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNetclassification with deep convolutional neural networks. InProc. NIPS, pages 1097–1105, Lake Tahoe, Nevada, USA,2012.
[63] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed,D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich.Going deeper with convolutions.CoRR, abs/1409.4842,2014
[64] C. Zach, T. Pock, and H. Bischof. A duality based approachfor realtime tv-l1 optical flow. InProceedings of the 29thDAGM Conference on Pattern Recognition, pages 214–223,Berlin, Heidelberg, 2007. Springer-Verlag.
[65] Zhang, Bowen, et al. "Real-time action recognition with deeply transferred motion vector cnns." *IEEE Transactions on Image Processing* 27.5 (2018).
[66] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overviewof the H.264/AVC video coding standard,"IEEE Trans. Circuits Syst.Video Techn., vol. 13, no. 7, pp. 560–576, Jul. 2003.
[67] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, andT. Wiegand, "High efficiency video coding (HEVC) text specificationdraft 10," inProc. JCT-VC, 2013, p. 91.
[68] G. Farnebäck, "Two-frame motion estimation based on polynomialexpansion," inProc. SCIA, Halmstad, Sweden, Jun. 2003, pp. 363–370.
[69] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracyoptical flow estimation based on a theory for warping," inProc. Eur.Conf. Comput. Vis. (ECCV), May 2004, pp. 25–36.

Bibliography

[70] Wang, Limin, Yu Qiao, and Xiaoou Tang. "Action recognition with trajectory-pooled deep-convolutional descriptors." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
[71] H. Wang and C. Schmid. Action recognition with improved trajectories. In ICCV, 2013.
[72] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for actionrecognition. IJCV, 103(1), 2013.
[73] J. Sánchez, F. Perronnin, T. Mensink, and J. J. Verbeek. Image classification with the Fisher vector: Theory and practice. IJCV, 105(3), 2013.
[74] H. Wang, A. Kl ̈aser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition.IJCV, 103(1), 2013.
[75] Ji, Shuiwang, et al. "3D convolutional neural networks for human action recognition." *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012)
[76] TRECVID dataset [Online]. Available: http://www-nlpir.nist.gov/projects/trecvid/
[77] Tran, Du, et al. "Learning spatiotemporal features with 3d convolutional networks." *Proceedings of the IEEE international conference on computer vision*. 2015.
[78] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, andL. Fei-Fei. Large-scale video classification with convolutional neuralnetworks. InCVPR, 2014.
[79] Varol, Gül, Ivan Laptev, and Cordelia Schmid. "Long-term temporal convolutions for action recognition." IEEE transactions on pattern analysis and machine intelligence, 2017.
[80] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, andL. Fei-Fei, "Large-scale video classification with convolutionalneural networks," inCVPR, 2014.
[81] V. Kantorov and I. Laptev, "Efficient feature extraction, encoding,and classification for action recognition," inCVPR, 2014.
[82] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracyoptical flow estimation based on a theory for warping," inECCV,2004
[83] Malik Tubaishat, Yi Shang, Hongchi Shi "Adaptive Traffic Light Control with Wireless Sensor Networks", Proceedings of IEEE consumer, 2007.
[84] Yen-Lin Chen, Bing-Fei Wu, Hao-Yu Huang, Chung-Jui Fan „A Real-Time Vision System for Nighttime Vehicle Detection and Traffic Surveillance", IEEE Transactions on Industrial Electronics Volume: 58, Issue: 5, May 2011.
[85] Jun-Wei Hsieh, Yung-Sheng Chen, Wen-Fong Hu „Automatic Traffic Surveillance System for Vehicle Tracking and Classification" , IEEE Transactions on Intelligent Transportation Systems Volume: 7, Issue: 2, June 2006.
[86] Tarik Taleb,Member, IEEE, Abderrahim Benslimane,Senior Member, IEEE, and Khaled Ben Letaief,Fellow, IEEEToward an Effective Risk-Conscious and Collaborative Vehicular Collision Avoidance System.
[87] Stauffer C, Grimson W. "Adaptive background mixture models for real-time tracking". Proc IEEE Conf on Comp Vision and Patt Recog (CVPR 1999) 1999; 246-252.
[88] Power, P. Wayne, and Johann A. Schoonees. "Understanding background mixture models for foreground segmentation." Proceedings image and vision computing New Zealand. Vol. 2002. 2002.
[89] Soille P. (1999) Erosion and Dilation. In: Morphological Image Analysis. Springer, Berlin, Heidelberg. http://doi-org-443.webvpn.fjmu.edu.cn/10.1007/978-3-662-03939-7_3.
[90] Suzuki, Satoshi. "Topological structural analysis of digitized binary images by border following." Computer Vision, Graphics, and Image Processing 30.1 (1985): 32-46.
[91] Ren, Mingwu, Jingyu Yang, and Han Sun. "Tracing boundary contours in a binary image." Image and vision computing 20.2 (2002): 125-131.
[92] Chow, Louis R., et al. "A new dynamic approach for finding the contour of bi-level images." CVGIP: Graphical Models and Image Processing 56.6 (1994): 507-509.

[93] Kirkpatrick, David G., and Raimund Seidel. "The ultimate planar convex hull algorithm?." SIAM journal on computing 15.1 (1986): 287-299.
[94] Chan, Timothy M. "Optimal output-sensitive convex hull algorithms in two and three dimensions." Discrete & Computational Geometry 16.4 (1996): 361-368.
[95] Graham, Ronald L., and F. Frances Yao. "Finding the convex hull of a simple polygon." Journal of Algorithms 4.4 (1983): 324-331.
[96]. Zoran Zivkovic, Ferdinand van der Heijden,"Efficient adaptive density estimation per image pixel for the task of background subtraction", Faculty of Science, University of Amsterdam, The Netherlands, 2004
[97]. Andrew Temlyakov, Brent C. Munsell, Jarrell W. Waggoner, Song Wang, "Two Perceptually Motivated Strategies for Shape Classification".
[98]. T.Y.Zhang, C.Y.Suen, "A Fast Parallel Algorithm for Thining Digital Patterns".
[99]. Indriyati Atmosukarto, Bernard Ghanem, Shaunak Ahuja, Karthik Muthuswamy, Narendra Ahuja, "Automatic Recognition of Offensive Team Formation in American Football Plays", 2013.
[100]. Maria Andersson, Luis Patino, Gertjan J. Burghouts, Adam Flizikowski, Murray Evans, David Gustafsson, Henrik Petersson, Klamer Schutte, James Ferryman, "Activity Recognition and Localization on a Truck Parking Lot", 2013.
[101]. Norsk Kennel Klub (NKK). "European Dog Show 2015 - Day 3." Online video clip. YouTube. YouTube, Sep 6, 2015.
[102]. Canal de PES10vsFIFA10. "Mujer atacada por manada de perros callejeros." Online video clip.YouTube. Youtube, Jul 20, 2010.
[103] Varol, G.; Laptev, I.; Schmid, C. Long-term temporal convolutions for action recognition. IEEE Trans. Pattern Anal. Mach. Intell. 2018, 40, 1510–1517.
[104] Kantorov, V.; Laptev, I. Efficient feature extraction, encoding and classification for action recognition. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 2593–2600.
[105] Blunsden, S.; Fisher, R.B. The BEHAVE video dataset: Ground truthed video for multi-person behavior classification. Ann. BMVA 2010, 4, 1–12.
[106] Patino, L.; Cane, T.; Vallee, A.; Ferryman, J. PETS 2016. Dataset and challenge. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 1–8.
[107] Rousseeuw, Peter J.; Hubert, Mia (2011), "Robust statistics for outlier detection", Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1 (1): 73–79.
[108] Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv 2017. arXiv:1704.04861.
[109] Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and<0.5 MB model size. arXiv 2016. arXiv:1602.07360.
[110] Advanced Convolutional Neural Networks. Available online: https://www.tensorflow.org /tutorials/images/ deep_cnn (accessed on 12 September 2018).
[111] Krizhevsky, A. Cuda-Convnet. Available online: https://code.google.com/archive/p/cuda-convnet/ (accessed on 12 September 2018).
[112] Soomro, K.; Zamir, A.R.; Shah, M. UCF101. A dataset of 101 human actions classes from videos in the wild. arXiv 2012. arXiv:1212.0402.
[113] UCSD Anomaly Detection Dataset. Available online: http://www.svcl.ucsd.edu/projects /anomaly/dataset. html (accessed on 12 September 2018).
[114] Kingma, D.P.; Adam, J. A method for stochastic optimization. arXiv 2014. arXiv:1412.6980.

[115] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 (2014): 1929-1958.
[116] Fortun, D.; Bouthemy, P.; Kervrann, C. Optical flow modeling and computation: A survey. Comput. Vis. Image Underst. 2015, 134, 1–21.
[117] Zhang, T. MoWLD: A robust motion image descriptor for violence detection. Multimed. Tools Appl. 2017, 76, 1419–1438.
[118] Cui, X.; Liu, Q.; Gao, M.; Metaxas, D.N. Abnormal detection using interaction energy potentials. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 20–25 June 2011; pp. 3161–3167.
[119] De Jong, M.; Joss, S.; Schraven, D.; Zhan, C.; Weijnen, M. Sustainable smart resilient low carbon eco knowledge cities; making sense of a multitude of concepts promoting sustainable urbanization. J. Clean. Prod. 2015, 109, 25–38.
[120] Juan, I.E.; Juan, M.; Barco, R. A low-complexity vision-based system for real-time traffic monitoring. IEEE Trans. Intell. Transp. Syst. 2017, 18, 1279–1288.
[121] Mohammad, R.; Sami, F. Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures. Autom. Constr. 2012, 22, 567–576.
[122] Bermejo Nievas, E.; Deniz, O.; Bueno, G.; Sukthankar, R. Violence detection in video using computer vision techniques. In Proceedings of the International Conference on Computer Analysis of Images and Patterns, Seville, Spain, 29–31 August 2011; pp. 332–339.
[123] Partha Pratim, R.; Mukherjee, M.; Shu, L. Internet of things for disaster management: State-of-the-art and prospects. IEEE Access 2017, 5, 18818–18835.
[124] Bautista-Durán, M.; García-Gómez, J.; Gil-Pita, R.; Mohíno-Herranz, I.; Rosa-Zurera, M. Energy-Efficient Acoustic Violence Detector for Smart Cities. Int. J. Computat. Intell. Syst. 2017, 10, 1298–1305.
[125] Hadjkacem, B.; Ayedi, W.; Abid, M.; Snoussi, H. A new method of video-surveillance data analytics for the security in camera networks. In Proceedings of the IEEE International Conference on Internet of Things, Embedded Systems and Communications IINTEC 2017, Gafsa, Tunisia, 20–22 October 2017; pp. 140–145.
[126] Mabrouk, A.B.; Ezzeddine, Z. Abnormal behavior recognition for intelligent video surveillance systems: A review. Expert Syst. Appl. 2018, 91, 480–491.
[127] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. Proc. IEEE 1998, 86, 2278–2324.
[128] Lee, Y.; Tsung, P.; Wu, M. Technology trend of edge AI. In Proceedings of the IEEE International Symposium on VLSI Design, Automation and Test, Hsinchu, Taiwan, 16–19 April 2018; pp. 1–2.
[129] Saif, S.; Tehseen, S.; Kausar, S. A survey of the techniques for the identification and classification of human actions from visual data. Sensors 2018, 18, 3979.
[130] Nam, J.; Alghoniemy, M.; Tewfik, A. Audio-visual content-based violent scene characterization. In Proceedings of the 1998 International Conference on Image Processing, Chicago, IL, USA, 7 October 1998; pp. 353–357.
[131] Clarin, C.; Dionisio, J.; Echavez, M.; Naval, P. DOVE: Detection of movie violence using motion intensity analysis on skin and blood. PCSC 2005, 6, 150–156.
[132] Chen, L.-H.; Su, C.-W.; Hsu, H.-W. Violent scene detection in movies. IJPRAI 2011, 25, 1161–1172.
[133] Giannakopoulos, T.; Makris, A.; Kosmopoulos, D.; Perantonis, S.; Theodoridis, S. Audio-visual fusion for detecting violent scenes in videos. In Proceedings of the Hellenic Conference on Artificial Intelligence, Athens, Greece, 4–7 May 2010; pp. 91–100.
[134] Davis, J.W.; Bobick, A.F. The representation and recognition of human movement using temporal templates. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, 17–19 June 1997; pp. 928–934.

[135] Laptev, I. On space-time interest points. Int. J. Comput. Vis. 2005, 64, 107–123.
[136] Chen, M.; Hauptmann, A. MoSIFT: Recognizing Human Actions in Surveillance Videos; Tech. Rep.; Carnegie Mellon University: Pittsburgh, PA, USA, 2009.
[137] Lowe, D. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. 2004, 60, 91–110.
[138] Fei-Fei, L.; Perona, P. A Bayesian hierarchical model for learning natural scene categories. In Proceedings of the Conference on Computer Vision and Pattern Recognition CVPR, San Diego, CA, USA, 20–26 June 2005; pp. 524–531.
[139] Peng, X.; Wang, L.; Wang, X.; Qiao, Y. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. Comput. Vis. Image Underst. 2016, 150, 109–125.
[140] Vapnik, V. The Nature of Statistical Learning Theory; Springer: New York, NY, USA, 1995.
[141] Andersson, M.; Patino, L.; Burghouts, G.J.; Flizikowski, A.; Murray, E.; Gustafsson, D.; Petersson, H.; Schutte, K.; Ferryman, J. Activity recognition and localization on a truck parking lot. Adv. Video Signal Based Surveill. 2013, 10, 263–269.
[142] Zivkovic, Z. Improved adaptive Gaussian mixture model for background subtraction. In Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, UK, 26 August 2004; Volume 2, pp. 28–31.
[143] Blackman, S.; Popoli, R. Design and Analysis of Modern Tracking Systems; Artech House: Boston, MA, USA, 1999.
[144] Wang, H.; Klaser, A.; Schmid, C.; Liu, C.-L. Action recognition by dense trajectories. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 20–25 June 2011; pp. 3169–3176.
[145] Wang, H.; Kläser, A.; Schmid, C.; Liu, C.-L. Dense trajectories and motion boundary descriptors for action recognition. Int. J. Comput. Vis. 2013, 103, 60–79.
[146] Rota, P.; Conci, N.; Sebe, N.; Rehg, J.M. Real-life violent social interaction detection; a new benchmark. In Proceedings of the International Conference on Image Processing, Quebec City, QC, Canada, 27–30 September 2015.
[147] Farneback, G. Fast and accurate motion estimation using orientation tensors and parametric motion models. In Proceedings of the International Conference on Pattern Recognition, Barcelona, Spain, 3–7 September 2000; Volume 1, pp. 135–139.
[148] Hassner, T.; Itcher, Y.; Kliper-Gross, O. Violent flows: Real-time detection of violent crowd behavior. In Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops, Providence, RI, USA, 16–21 June 2012; pp. 1–6.
[149] Gao, Y.; Liu, H.; Sun, X.; Wang, C.; Liu, Y. Violence detection using Oriented Violent Flows. Image Vis. Comput. 2016, 48, 37–41.
[150] Ilg, E.; Mayer, N.; Saikia, T.; Keuper, M.; Dosovitskiy, A.; Brox, T. Flownet 2.0: Evolution of optical flow estimation with deep networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; Volume 2, pp. 1–6.
[151] Hui, T.-W.; Tang, X.; Loy, C.-C. LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 March 2018; pp. 8981–8989.
[152] Taylor, G.W.; Fergus, R.; LeCun, Y.; Bregler, C. Convolutional learning of spatio-temporal features. In Proceedings of the European Conference on Computer Vision, Heraklion, Greece, 5–11 September 2010; pp. 140–153.
[153] Donahue, J.; Hendricks, L.A.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K.; Darrell, T. Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 2625–2634.

[154] Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; Van Gool, L. Temporal segment networks: Towards good practices for deep action recognition. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 20–36.
[155] Herath, S.; Harandi, M.; Porikli, F. Going deeper into action recognition: A survey. Image Vis. Comput. 2017, 60, 4–21.
[156] Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Li, F. Large-scale video classification with convolutional neural networks. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1725–1732.
[157] Baccouche, M.; Mamalet, F.; Wolf, C.; Garcia, C.; Baskurt, A. Sequential deep learning for human action recognition. In Proceedings of the International Workshop on Human Behavior Understanding, Amsterdam, The Netherlands, 16 November 2011; pp. 29–39.
[158] Xu, D.; Ricci, E.; Yan, Y.; Song, J.; Sebe, N. Learning Deep Representations of Appearance and Motion for Anomalous Event Detection. arXiv 2015. arXiv:1510.01553.
[159] Dai, Q.; Zhao, R.W.; Wu, Z.; Wang, X.; Gu, Z.; Wu, W.; Jiang, Y.G. Detecting Violent Scenes and Affective Impact in Movies with Deep Learning. In Proceedings of the MediaEval 2015 Workshop, Wurzen, Germany, 14–15 September 2015.
[160] Sudhakaran, S.; Lanz, O. Learning to Detect Violent Videos using Convolutional Long Short-Term Memory. In Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance, Lecce, Italy, 29 August–1 September 2017.
[161] Zhou, P.; Ding, Q.; Luo, H.; Hou, X. Violent Interaction Detection in Video Based on Deep Learning. J. Phys. Conf. Ser. 2017, 844, 012044.
[162] Kuehne, H.; Jhuang, H.; Stiefelhagen, R.; Serre, T. Hmdb51, a large video database for human motion recognition. High Perform. Comput. Sci. Eng. 2013, 12, 571–582.
[163] Zhang, T.; Yang, Z.; Jia, W.; Yang, B.; Yang, J.; He, X. A new method for violence detection in surveillance scenes. Multimed. Tools Appl. 2016, 75, 7327–7349.