Online Video Analysis for Abnormal Event Detection and Action Recognition

by

Marcial Roberto Leyva Fernandez

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department Of Computer Science

August 2018

Contents

List of Tables

List of Figures

Acknowledgments

To Amparo, the flower my land awarded me. My devotion from A to Z for her radiance and love.

I would like to express my most profound gratitude to Victor Sanchez and Chang-Tsun Li, who guided me during my PhD studies with patience and respect. Their experience and knowledge were crucial aspects to improve my research skills.

I dedicate this thesis as well to my parents Rosario Fernandez, and Abelardo Leyva whose love inspired me to achieve this goal and this effort also belong to them. My most profound gratitude to both.

I would also like to thank Dr. Abhir Bhalerao, Dr. Alexandros Iosifidis and Dr. Theo Damoulas for their valuable advice during my studies.

And last but never the least, to my colleagues Xufeng Lin, Ning Jia, Xin Guan, Qiang Zhang, Bo Wang, Shang Lin, Ruizhe Li, Alaa Khadidos and Miguel Carbonero for their kindness, support, and enthusiasm.

Declarations

I hereby declare that this dissertation entitled ***Online Video Analysis for Abnormal Event Detection and Action Recognition*** is an original work and has not been submitted for a degree or diploma or other qualification at any other University.

Coventry, United Kingdom.

Abstract

Automatic video surveillance has become one of the most active research areas in computer vision. Its applications are vast; these include security purposes, patient monitoring and law enforcement. Considering that millions of cameras operate all over the world, human surveillance is impractical for many reasons. Perhaps the most important reason is that strictly speaking, we require one person to monitor one camera. This monitoring is not only unrealistic but also inefficient because we cannot have a person 24/7 observing a scene. Even if that would be possible, fatigue and distractions might deter its efficiency.

The main challenge of video surveillance is that it requires online processing (no-cumulative delay process) for practical scenario purposes. The reason is that the system's response should be given immediately after the event occurred. If this time requirement is not satisfied, the system will end up warning the operators minutes or hours later. Then, the system's response will be impractical for some events (e.g. crimes, accidents and fires) where the response times are critical.

Although many methods have been developed for video surveillance, there is very little in terms of online-based methods. The lack of online approaches has been because there is a trade-off between accuracy in detecting events and computational complexity. The objective of this thesis is to minimise the gap of the speed-accuracy trade-off. To this end, this thesis proposes: (I) multi-source motion extraction to boost accuracy and expand the type of events to be detected, (II) extract few but high descriptive features via multi-scale extraction with perspective compensation, and (III) four fast binary-based video descriptors.

The main findings of this thesis are as follows: First, multi-scaled perspective features reduce computational times meeting online requirements in abnormal event detection. Second, binary video features achieve competitive accuracy in action recognition compared with existing features while drastically outperform them in terms of computational complexity.

In conclusion, first, by carefully selecting the spatio-temporal regions to process video data significantly improves accuracy and at the same time reduces computational times to detect abnormal events. Second, binary video features can compete with existing features by selecting a limited number of descriptive spatio-temporal symmetric regions. Finally, the findings of this thesis could benefit all those video applications that require real-time or online processing times.

Chapter 1

Introduction

In recent years remarkable advances in video technology have to lead striking computer vision applications. Among these is video content analysis of which the two most known categories are action recognition and video surveillance. Still, enormous challenges are under research to make state-of-the-art techniques feasible for practical applications. This aspect is mainly because some computer vision tasks, such as video surveillance, require processing data quite fast to be functional in real scenarios.

1.0.1 Practical Applications Needs for Video Surveillance

Practical applications are methods/techniques whose computational demands are reasonable for large or medium scale operation. This thesis is focused on minimising computational times of crucial video processing steps for for that end.

Fast video processing is mandatory if we want to detect events immediately as they occur. Otherwise, long processing times will cause the system give its response minutes, hours, or any time unit **after** the actions have already passed. This detection is impractical for instance in a robbery, where the operators probably **need** to be aware in real time of this sort of event.

This thesis addresses fast video processing. Processing video in this light has notable advantages to build *practical applications* for video surveillance, as we will demonstrate during the subsequent chapters.

1.0.2 Justification of my Thesis

My motivation is based on the ever-growing need for automatic video surveillance. Just in recent years, the number of cameras fitted in public scenarios in this country

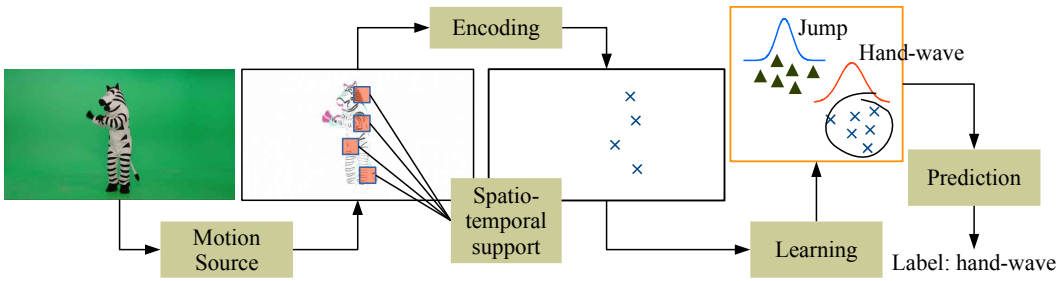


Figure 1.1: Typical pipeline for video understanding.

nearly doubled [3]. As more millions of cameras are added every year around the globe, this dramatic change requires efficient online video processing techniques. Hence the motivation for developing fast processing techniques for automatic event detection.

1.0.3 Automatic Video Surveillance Challenges

Although extensive work exists, the majority lacks online performance. Many authors trust that an efficient implementation of their proposals (e.g. code language, specific purpose hardware, computational resources) would one day make possible developing applications for real scenarios. This thesis also discloses such assumption, which transforming offline into online performing methods is not a trivial task and demands elaborate analysis.

Video analysis for online performance has to be such that requires low computational resources to be practical as some authors explicitly pointed out [1, 4]. We cannot demand the computational power, e.g. a server, to detect events in one camera only. Subsequent chapters demonstrate how challenging it is to meet online requirements with competitive accuracy and low computational cost demands.

Reducing computational costs without significant accuracy loss is very challenging to achieve and is the primary motivation for this thesis.

1.1 Video Analysis

Video analysis is a category of computer vision that focuses on extracting data from the video sequence to its further comprehension. Its applications are vast and unaccountable, to name few, abnormal event detection, objects tracking, abandoned luggage and action recognition [5]. To this end, authors employ double precision features [6], clustering, statistical models [7] and more recently neural networks [8].

Figure 1.1 depicts a typical pipeline to analyse video. The steps of the pipeline are similar for many video applications, and commonly we find them consecutively as:

1. Motion source: refers to create the input space to be processed. In practice, it can be the raw video data as pixels intensities or motion transformations, e.g. optical flow or binary foreground.

2. Spatio-temporal support detection: refers to the spatio-temporal region extent to be analysed from the motion source. These regions can be either the whole sequence sampled at each spatio-temporal location (no detection) or specific ones detected, for instance, by interest points detectors.

3. Encoding: refers to the spatio-temporal support mapping into a new feature space. This step is also known as local feature extraction or descriptor encoding.

4. Learning: refers to train the response of a model with the mapped features of the encoding step.

5. Predicting: refers to the model's response to the newly observed features.

The system response, e.g. a label, is the interpretation of one event. This label given by the model could mean, for instance, the presence or absence of a particular object, the occurrence of an accident or the classification of a set of human movements (e.g. walking).

1.2 Abnormal Event Detection

One of the most useful applications of video analysis is abnormal event detection. The vast gamma of applications in this area, e.g. facility protection, vandalism deterrence, patient monitoring, accident detection, public safety, has resulted in significant advances to the state-of-the-art.

Abnormal event detection methods have been shown to be highly efficient to detect unusual events without *a priori* knowledge of them [3, 9]. This property is one of their most valuable advantages since the number of anomalous events can be enormous [3, 9].

Figure 1.2: Illustrations from [10], (Top) a sequence of movements given to the machine to learn usual behaviour. i.e. normal activity. (Bottom) a sequence of movements never given to the machine and consistently different from the learned ones, therefore anomalous.

1.2.1 Abnormal Event Definition

The first step is to define what an *abnormal*, *unusual* or *anomalous* event is. Henceforward, an abnormal event is a sequence of actions that rarely or never have occurred in a video sequence. Let us consider the events that never happened before; one can easily see that those events cannot be explicitly defined. [3]. Hence some systems consider unusual events as the events that occasionally occur [9]. In this case, we deem a spatio-temporal region in a video sequence anomalous if this region has a low probability of being composed of previous observations. Figure 1.2 illustrates this concept.

To exemplify more in detail the *abnormal event concept*, let us consider a scene of one store, where customers perform three sequences of movements. These are, picking up items, paying and leaving the store. Although the number of possible interactions is uncountable and strictly speaking two actions are not entirely the same, we can say that is **normal** the occurrence of these three sequences of movements, and we expect that further actions are similar to them. For instance, very frequently customers take and place items in a basket in many different but alike ways. Now imagine that two burglars broke into the store, beat the cashier and force the customers to lie on the ground. For this event, it is difficult to establish how to use the observations of picking up items to describe this new sequence of events. Therefore, we interpret this *new* observed event as abnormal.

We can provide a formal definition of abnormal event detection based on evidence:

> *The modelled spatio-temporal regions should provide enough evidence that what we are seeing has already occurred in the past. Otherwise, we consider it as abnormal.*

1.2.2 Advantages of Abnormal Event Detection

Detecting unusual events under the *abnormal event concept* perspective has significant advantages. As a counterexample, using specific event detection approach [3], we have to feed the computer with samples of actual robberies. The machine should learn how that event looks like and proceed to detect it. This logic presents two main drawbacks.

One is that the method will learn to identify only that event. As we will see in Chapter 4, there are very intricated scenes where we can not even see the actual robbery on the footage but the people running as its cause. Then the machine should learn a cascade of events to detect the theft. We will see that such sequences are very complicated to be rigidly refined.

Two, what could happen in the scenes has no boundaries. As Chapter 4 illustrates with a particular example, those scenes might become very intricate. One of them involves fighting, followed by people running in panic and on the very same scene vehicles set on fire. It is hard to define this sequence of events, and eventually, we will require finding similar sequences to feed the machine to detect such events.

Abnormal event detection does not require manually finding of such cascades of events. **No explicit event definition** of the actions that could happen in a scene is its most important advantage over specific event detection.

1.2.3 Disadvantages of Abnormal Event Detection

Abnormal event detection has drawbacks, of course, I discuss the two most important. There are no precise rules to define how many observations are sufficient to build the model that detects the events. This imprecision presents a problem: the method cannot learn all the time, it is impossible. We have to understand that there is no infinite memory to store all the previously seen observations. This limitation forces us to update the model with the new inputs, but updating it can cause forgetting what the model has already learnt [4]. Thus, under this scope, the inference

method is limited to detect abnormal events with *recent* knowledge because it has forgotten what has already learnt.

Another complication is that meeting online requirements is a high challenge for video surveillance. Although at first glance this seems just a matter of reducing computational times, abnormal event detection is *time constrained*. This aspect is challenging because from the vast catalogue of computer vision techniques, (e.g. tracking, segmentation, filtering and encoding) hardly any is aimed to be online executed.

We stress that any technique of the abnormal event detection system should require the order of few milliseconds for its processing or else be discarded as practical in the light of online processing.

1.2.4 Challenges of Abnormal Event Detection

To build a practical event detection system requires addressing the *time constraint*. To illustrate its difficulties, we present few examples.

If we create an abnormal event detection system, which employs as motion source (first point listed in Section 1.1) dense optical flow, its calculation time for a QCIF-frame[1] will be c.a. 22ms. This system needs processing frames at 30 Frame Per Second (FPS) [2] (or 33ms) to be online performing. Then 22ms from the motion source represents 2/3 of the total system's execution time, and this is only one of the multiple steps to be performed by the system. This motion source leaves practically no room for another processing step, and thus we can not use it.

The same time aspect emerges when extracting features (second point listed in Section 1.1), for instance, 3DSIFT [3] requires c.a. 220ms per spatio-temporal location. As Section 2.2.2 discusses, some methods might require extracting thousands of features per frame making nearly impossible to employ some visual descriptors for online abnormal event detection. Chapter 4 shows that the rest of the processing steps listed in Section 1.1 are equally challenging regarding processing times.

Meeting online requirements is not a trivial task; it requires a series of strategies that require small fractions of time. This time constraint implies that all the individual steps listed in Section 1.1 must meet online requirements individually. These aspects demand elaborate formulations to achieve competitive accuracy and are the very basis of this thesis.

[1] This video resolution corresponds to 176×120 frame size.
[2] broadly accepted standard video surveillance frame rate
[3] 3DSIFT generates a 2048-dimensions vector, Scovanner et al. [11]

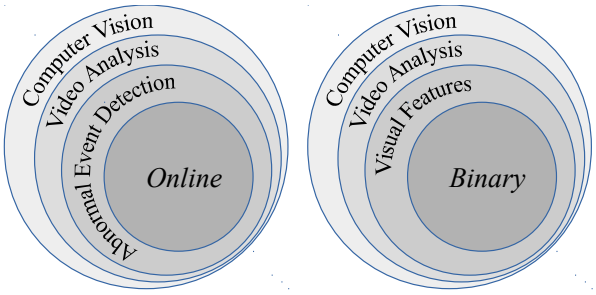


Figure 1.3: Our contributions summarised. (left) Online based abnormal event detection. (right) binary-based features for video analysis.

1.3 Our Contributions

This thesis proposes a set of strategies aimed at fast video analysis. These are specifically designed to detect events with online performance. Figure 1.3 conceptually illustrates our contributions and are as follows:

- A set of strategies to detect abnormal events in online fashion. One of the core aspects is the compact set of features. The extraction of few but descriptive features is essential to reduce computational times.

  We need to carefully select regions in the scene at the right scale with significant motion otherwise the amount of data would increase drastically. An immense amount of data would make impossible online abnormal event detection. To this address this problem we propose an efficient technique, which is using variable-sized support regions and foreground cell activation to process the motion source.

  The proposed technique has three advantages. (I) The feature extraction from specific regions avoids extracting thousands of features compared with existing methods. (II) Employing variable-sized support regions avoids multiscale processing, which is one of the principal reasons behind long processing times. (III) It discards motionless areas. These significantly produce noisy features hindering the inference mechanism accuracy to detect events.

- Four binary-based descriptors for fast video analysis. (I) Two of them encode support regions using summed video volume comparisons. (II) The other two binarise the trajectories generated by optical flow. All descriptors require the order of milliseconds to be computed. Moreover, they achieve competitive accuracy as tested on action recognition.

- A Fisher Vectors variant for binary data. Following the idea of Fisher Vectors for double precision features, a binary-based version projects binary data into high dimensional spaces. The results significantly outperform traditional models regarding accuracy.

1.4 Thesis Organization

This thesis is organised in the following manner and comprises:

- Chapter 2: a review of abnormal event detection, visual descriptors and important concepts to understand this thesis.
- Chapter 3: a graph-based method to detect abnormal events.
- Chapter 4: compact set of features for online abnormal event detection.
- Chapter 5: binary-based descriptors tested on action recognition.
- Chapter 6: the relevance of this thesis and open problems of abnormal event detection and action recognition.

1.5 Publications

The list of publications associated with this thesis is the following:

- Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. Video anomaly detection based on wake motion descriptors and perspective grids. In *IEEE International Workshop on Information Forensics and Security (WIFS), 2014*, pages 209–214, Dec 2014. doi: 10.1109/WIFS.2014.7084329
- Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. A fast binary pair-based video descriptor for action recognition. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4185–4189, Sept 2016. doi: 10.1109/ICIP.2016.7533148
- Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. Fast binary-based video descriptors for action recognition. In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, Nov 2016. doi: 10.1109/DICTA.2016.7797041

- Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. The lv dataset: a realistic surveillance video dataset for abnormal event detection. In *2017 International Workshop on Biometrics and Forensics*, Mar 2017

- Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. Video anomaly detection with compact feature sets for online performance. *IEEE Transactions on Image Processing*, 26(7):3463–3478, July 2017. ISSN 1057-7149. doi: 10.1109/TIP.2017.2695105

This page intentionally left blank.

Chapter 2

Literature Review

This chapter presents important concepts to understand our contributions. To this end, we first describe existing work and fundamental theory. This chapter addresses in three separate sections the previous work related to this thesis. Firstly, section 2.1 comprises essential concepts required through subsequent chapters. This section provides only theoretical aspects and definitions. Section 2.2 describes the relevant work of abnormal event detection. Section 2.3 describes spatio-temporal features commonly used for both abnormal event detection and action recognition.

2.1 Computer Vision and Machine Learning

In this thesis, to clearly appreciate the contribution of this thesis, it is essential to review concepts from machine learning and computer vision. These will be used extensively in subsequent chapters. Since the thesis' subject is video understanding, we start with features for images and videos. This category is the cornerstone of this thesis, and most of its contributions lie therein.

2.1.1 Visual Features

In computer vision, a visual **feature** is a vector x that represents a **support region**. The support region is a region comprising N pixels. The support region is mapped to a new dimensional space to create a feature. Thus, the mapping is as $\mathbb{R}^2 \rightarrow \mathbb{R}^D$ for images and $\mathbb{R}^3 \rightarrow \mathbb{R}^D$ for videos. We commonly name this mapping as feature extraction. Figure 2.1 depicts feature extraction methodology for image and video. For the case of video, the support region is called Spatio-Temporal Support Regions (STSR) also known as video volume.

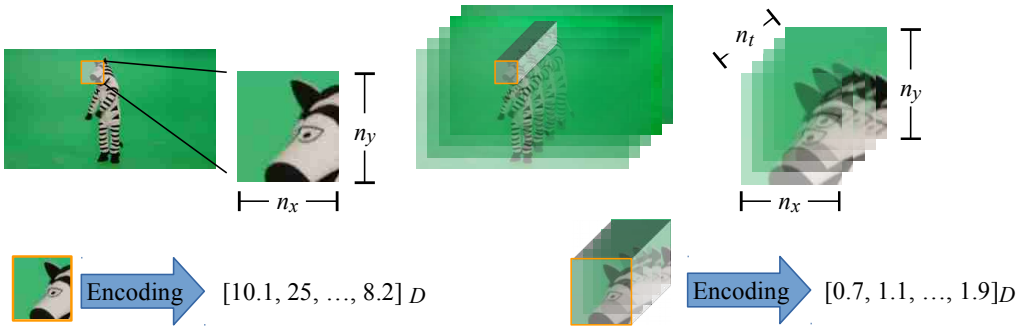


Figure 2.1: Image local support region and encoding $n_x \times n_y$ (left) and video $n_x \times n_y \times n_t$ (right).

The support region can be the entire image/video, but most of the times it is just a small local region around a pixel. For this reason, we define as **local feature** as the feature extracted from a small region.

2.1.2 Binary Features

Binary features are visual features whose domain is the base-2 numerical system. Thus, the binary feature vector $x = \{x_1, x_2, \ldots x_i, \ldots, x_D\}$ comprises D number of bits, i.e., $x_i \in \{0, 1\}$.

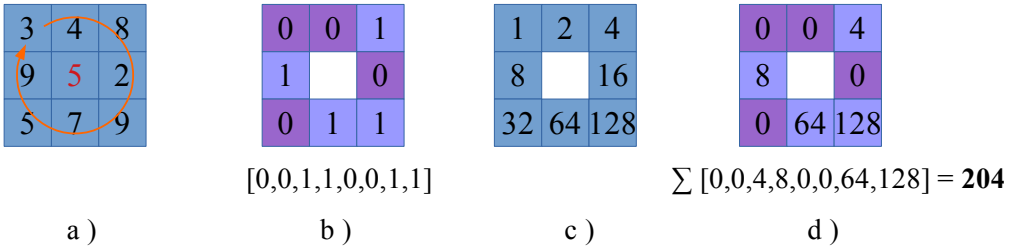


Figure 2.2: a) Given a support region, b) the centre pixel is compared with its neighbourhoods. c) Each bit is weighed by power two to generate d) an 8-digit binary number .

To extract a binary feature vector from an image or video, Ojala et al. [17] proposed to make pixel comparisons. The proposed method, which is called Local Binary Pattern (LBP), requires selecting a support region and its centre. For every pixel, we compare the centred pixel with each of its eight neighbours (on its top-left, middle-left, bottom-left, top-right, etc.). We perform this process along a circle, i.e., clockwise or counter-clockwise (see Figure 2.2). The LBP produces a feature vector by setting the x_i bit to 1 if the centre pixel's value is greater than the neighbour's value, and otherwise $x_i = 0$. This method gives an 8-digit binary number.

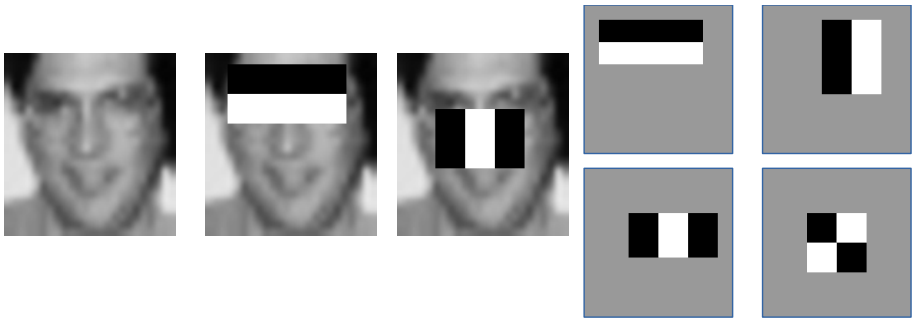


Figure 2.3: (left) The first and second features selected by AdaBoost. The two features are shown overlayed on a training face. (right) The Haar features provided to the classifier. These are shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels which lie within the black rectangles.

The LBP principle is generating binary features via pixel comparisons. This idea was reused by Viola and Jones [18] to detect human faces. They consider subregions of the support region instead of pixels. Thus, the **Viola-Jones detector** is a subregion comparison method. The Viola-Jones detector works as follows. A variant of the AdaBoost classifier [19] is fed with positive and negative sample images of human faces. The algorithm retrieves the features that provide the most accurate classifier. Figure 2.3 (right) shows the features that boost the classification. These correspond to **Haar-based patterns**.

The first retrieved feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalises on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose. Their proposed work demonstrates that analysing regions using Haar-based patterns is very efficient to detect human faces.

2.1.3 Interest Points

Interest Points commonly correspond to the edges and corners of an image or video. In the case of images, we call the regions Interest Point, while in videos, Spatio-Temporal Interest Point (STIP). To detect the STIPs, some detectors require convolving the image I with one or more filters g_k [20].

$$G = g_k * I \tag{2.1}$$

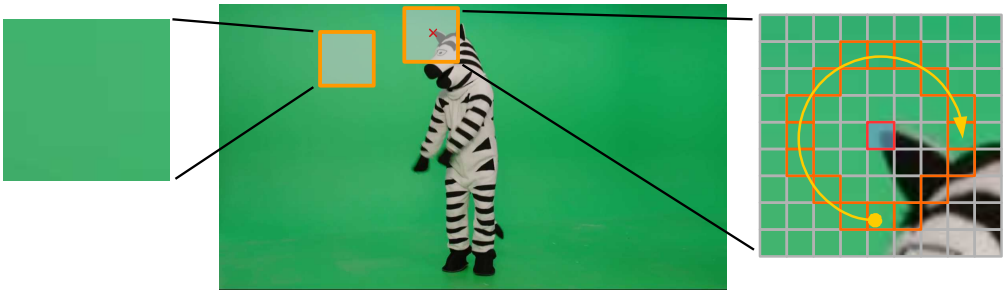


Figure 2.4: Homogeneous region (left) containing little information about the image. Interest point (right), a descriptive region comprising the zebra's ear detected by FAST corresponds to an Interest Point.

For every pixel in the image, we analyse its spatial surrounding. Some detectors, e.g. Fast Accelerated Segmentation Test (FAST) [21], do not require convolutions. However, other detectors e.g. Scale Invariant Feature Transformation (SIFT), need elaborate analysis of the pixel values when applying different filters.

Figure 2.4 depicts two regions of the image. The FAST interest point is that pixel whose intensity is greater or lesser than nine consecutive surrounding pixels. As we can observe from the image, the interest point comprises useful information of the image's subjects.

A Region of Interest (ROI) is a region that comprises one or more interest-points. Very commonly an *interest-point detector* detects this region, however, this is not always the case. For instance, this region could be the one comprising white and black pixels in Figure 2.4. In that case, the ROI contains the zebra. Thus the ROI is a region that is useful for a particular purpose. In subsequent sections, the ROI comprises the STSR where a specific action takes place, e.g., a robbery, an accident, etc.

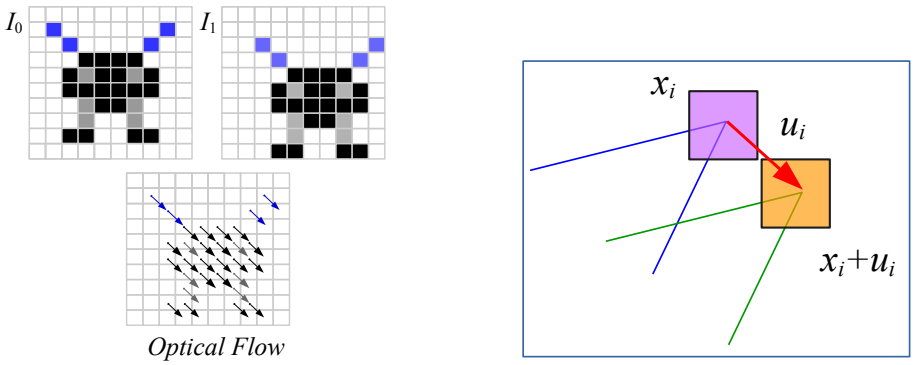
2.1.4 Optical Flow

In computer vision, the optical flow is a pattern to estimate movement between consecutive images or video frames. Due to the moving objects, the observed effect is pixels intensity changes [20]. Determining motion via the analysis of pixels intensity changes is therefore necessary. This process involves minimising the pixel differences of every pixel across frames. Given two consecutive frames I_0 and I_1, the goal is to estimate the vector u that generates the movement at location x_i and minimises the

sum of squared differences (SSD) as:

$$E_{SSD-OF}(u) = \sum_i \left(I_1(x_i + u) - I_0(x_i)\right)^2 . \tag{2.2}$$

This expression is commonly known as **Displacement Image Function**. Since the number of variables of u is larger than the number of measurements, the problem is undetermined. Hence, one of the classic approaches to solve this problem is by window matching in the local neighbourhood. This method usually involves a Taylor series expansion of the Displacement Image Function.



a) b)

Figure 2.5: a) Concept illustration of optical flow across consecutive images. b) Optical flow estimation via window matching. The corner in Frame I_0 is illustrated in blue while in I_1 in green.

Figure 2.5b) depicts the concept of estimating the displacement u. In this figure, the window located at x_i is displaced in I_0 to its new position in I_1 due to u.

2.1.5 Lucas-Kanade Optical Flow

Lucas and Kanade [22] proposed to perform Gradient Descent on the Displacement Image Function (Equation 2.2). To this end, we approximate the function by the Taylor series expansion as:

$$E_{LK-SSD}(u + \Delta u) = \sum_i \left(I_1(x_i + u + \Delta u) - I_0(x_i)\right)^2 \tag{2.3a}$$

$$\approx \sum_i \left(I_1(x_i + u) + J_1(x_i + u)\Delta u - I_0(x_i)\right)^2 \tag{2.3b}$$

$$= \sum_i \left(J_1(x_i + u)\Delta u - e_i\right)^2 , \tag{2.3c}$$

where J is the Jacobian at $(x_i + u)$ given by the gradient ∇:

$$J_1(x_i + u) = \nabla I_1(x_i + u) = \left(\partial_x \left(I_1\right), \partial_y \left(I_1\right)\right)(x_i + u) \tag{2.4}$$

and the intensity error is given by:

$$e_i = I_1(x_i + u) - I_0(x_i). \tag{2.5}$$

The gradient at a particular pixel location $x_i + u$ can be computed taking vertical and horizontal differences between pixels. The linear form of the incremental update to the SSD (Equation 2.3a) is called optical flow constraint and is as follows:

$$I_x u_x + I_y u_y + I_t = 0 \tag{2.6}$$

The I_x and I_y denote the spatial derivatives, and I_t the temporal derivative. The displacement components are u_x and u_y respectively. The *associated normal equation*[1] can be used to minimise Equation 2.3a

$$A\Delta u = b \tag{2.7}$$

where

$$A = \sum_i J_1^\top(x_i + u) J_1(x_i + u) \tag{2.8}$$

and

$$b = -\sum_i e_i J_1^\top(x_i + u) \tag{2.9}$$

are called the Hessian and gradient-weight residual vector, respectively. We can rewrite the matrices (Equations 2.8 and 2.9) as:

$$A = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}, \quad b = \begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix} \tag{2.10}$$

Finally the Lucas-Kanade optical flow is determined as:

$$\begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix} \tag{2.11}$$

The Lucas-Kanade optical flow method solves the problem of determining the displacement vector. In Figure 2.5b) the matching of the patch located at x_i gives, as a

[1] Linear Least Squares solution taken from [23]

result, the displacement u. One significant advantage of the Lucas-Kanade method is its computational time.

As we explained in previous sections, detecting abnormal events in this thesis should start from the large-scale operation needs as Adam et al. [1] suggest. The main reason behind the high complexity of some optical flow methods is that each pixel has to be located across frames, finding it requires commonly $O(n^2)$ computations for a window with n pixels.

The optical flow can be more accurate if we consider different scales, colour components, geometrical transformations or next and previous frame recalculation [24]. The complexity, in that case, would be $O(n^3)$ for each pixel, making the method very computationally expensive.

In the light of online processing, we do not consider optical flow methods that require specific purpose hardware, e.g. GPU, FPGA or multicore architectures for the sake of their high complexity. For example, some authors report speedup of nearly $+80\times$ using GPU [25]. Therefore, their proposed methods should have very long processing times when executed by single core processors. This drawback leads to seeking the best reported optical flow method regarding processing times.

Table 2.1 tabulates processing times to extract the optical flow by different methods. From this table, the FOLKI method seems to fit best the needs. However, an optimised Lucas-Kanade implementation (C++) has shorter processing times with competitive accuracy. Thus in future sections, we employ the Lucas-Kanade optical flow method.

Table 2.1: Processing time for optical flow methods per frame [24].

Method	Processing Time (seconds)	CPU/GPU	Language
2D-CLG [26]	844	GPU	Cuda/C++
Second-order prior [27]	14	GPU	Cuda/C++
Adaptive [28]	9.2	GPU	–
Aniso-Huber-L1 [25]	2	GPU	Cuda/C++
Horn&Schunck [29]	49	CPU - one core	Matlab
Lucas-Kanade [22]	11.9	CPU - one core	Matlab
FOLKI [30]	1.4	CPU - multicore	C++

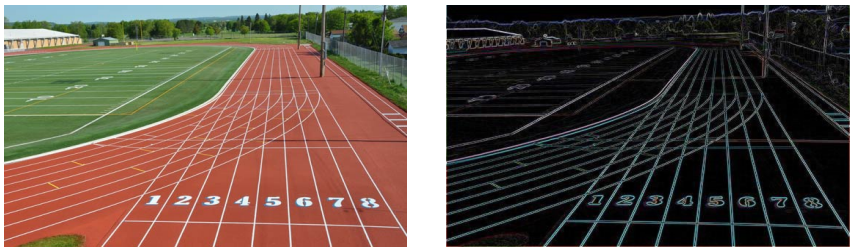


Figure 2.6: The image I (left) and its Gradient magnitude G (right)

2.1.6 Histogram of Oriented Gradients

One of the most successful features to capture local image information, proposed by Dalal et al. [31], is named Histogram of Oriented Gradients (HOG). This descriptor comprises horizontal G_x and vertical G_y gradient information (see Figure 2.6). To calculate the gradients, the Prewitt operator is convolved with the image I as:

$$G_x = [-1, 0, 1] * I \tag{2.12a}$$

$$G_y = [-1, 0, 1]^\top * I \tag{2.12b}$$

$$G = \sqrt{(G_x)^2 + (G_y)^2} \tag{2.12c}$$

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{2.12d}$$

To calculate the HOG feature x, we select a spatial support region. This spatial support region is a pixel window of size $n_x \times n_y$. For each pixel in the window, we bin the Gradient direction Θ into K number of bins and accumulate its magnitude. The direction binning is given by:

$$j : \underset{b_j}{\operatorname{argmin}} \ |\Theta_i - b_j| \tag{2.13}$$

where b_j is the bin, e.g., angles $\{0, \pi/4, \ldots, 2\pi\}$. This equation provides the bin b_j closest to the angle value of Θ_i. We accumulate the values of G_i using the index j of the vector x as:

$$x(j) = x(j) + G_i \tag{2.14}$$

Thus x has a dimension equal to the number of bins. After the binning, the vector x is L_1 normalised for scale invariance. Figure 2.7 depicts an example of HOG generation. We select the support region close to the number seven and zoom in twice to appreciate the pixel values for G and Θ. Notice that in the case of video,

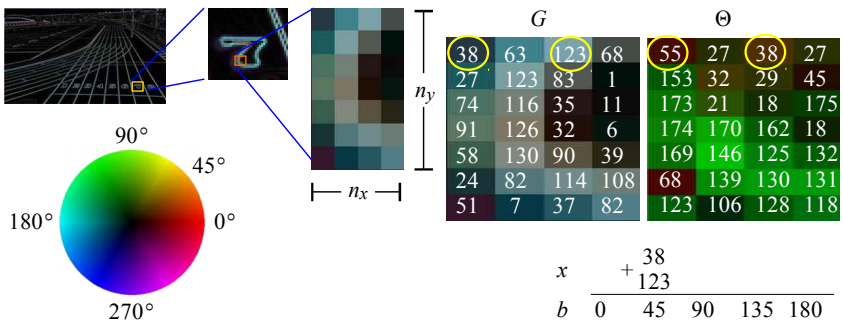


Figure 2.7: HOG descriptor example generation. The Orientation Θ is coloured using the model of the circle. Encircled in yellow, we add the magnitude of two degrees $(55, 38)$ to the bin 45, i.e., $38 + 123$.

it is possible to include the temporal gradient G_t and evaluate the azimuthal angle, this provides the 3-Dimensional Histogram of Gradients (3DHOG) features.

2.1.7 Histogram of Optical Flow

The Histogram of Optical Flow (HOF), proposed by Dalal et al. [31], captures local video information. They calculate this descriptor in the same way as HOG. However, the information source is the optical flow (see Figure 2.8). They use the optical flow's horizontal and vertical motion components to create the feature vector.

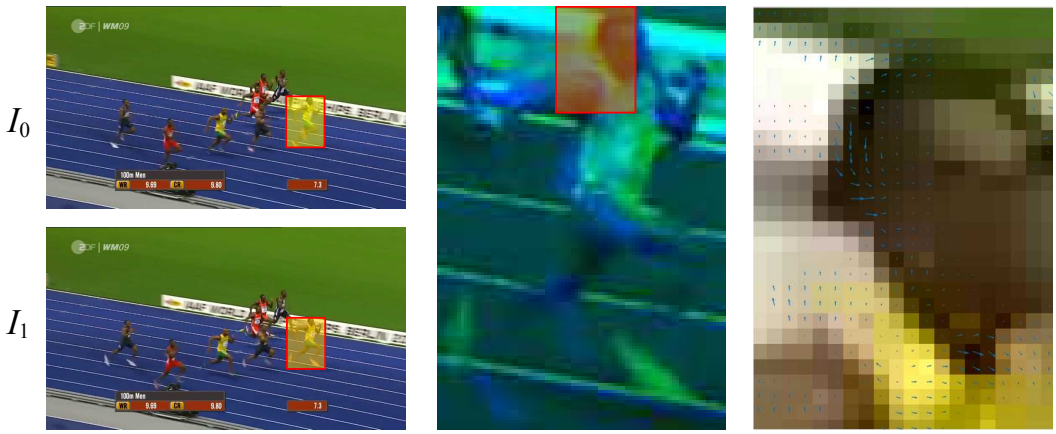


Figure 2.8: Optical Flow from two consecutive video frames (left). Highlighted in green and blue are the frame's temporal differences (centre). From a patch (right), we display the optical flow motion vectors.

Frames I_0 and I_1 generate the optical flow O. The magnitude M and orientation Θ of the motion components O_x and O_y are used to calculate the HOF

descriptor (see Figure 2.9). Those sources are given by:

$$M = \sqrt{(O_x)^2 + (O_y)^2} \tag{2.15a}$$

$$\Theta = \arctan\left(\frac{O_y}{O_x}\right) \tag{2.15b}$$

As we can see, these expressions are the same as the Gradient magnitude case of HOG. Generating the descriptor x, extracted from optical flow, follows the same procedure of binning the directions as:

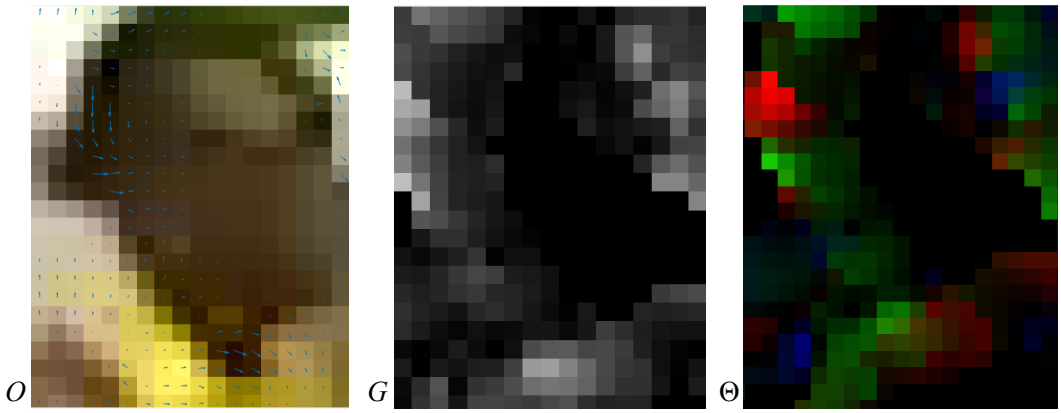


Figure 2.9: Optical Flow (left) magnitude (center) and orientation (right) required to extract the descriptor.

$$j : \underset{b_j}{\operatorname{argmin}} \ |\Theta_i - b_j| \tag{2.16a}$$

$$x(j) = x(j) + M_i \tag{2.16b}$$

Finally, the vector x is L_1-normalised. The HOF descriptor captures motion information via the optical flow magnitude and direction. We make use of this property in future sections to detect events in the video.

2.1.8 Bag Of Features

Bag of Features (BoF) is an algorithm to compute the features' distribution via feature matching [32]. The algorithm produces a vector given a set of features. Commonly this vector is a histogram. To generate such histogram, BoF requires two sets of features. In one set are the features $X = \{x_1, \dots x_i, \dots x_N\}$ from which we want to calculate the histogram h. In the second set the features to be matched. This second set is known as the bag or dictionary $D = \{d_1, \dots d_m, \dots d_M\}$.

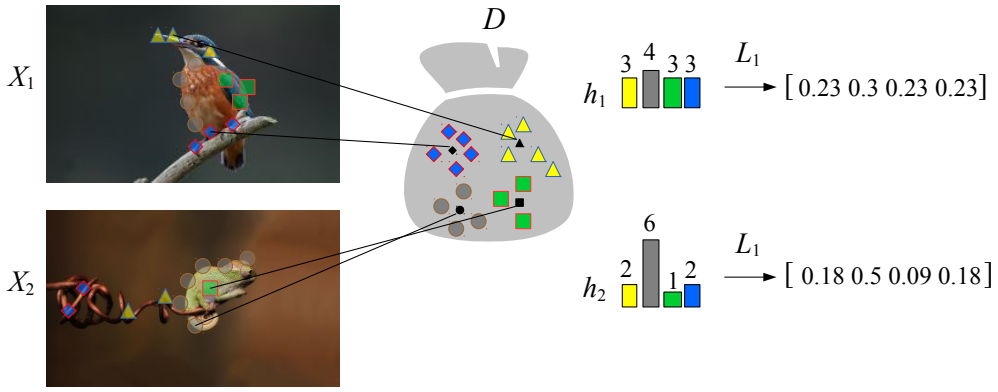


Figure 2.10: BoF alogorithm. Features from two images (X_1, X_2) are matched with D to create histograms (h_1, h_2).

The BoF requires to find the feature d_j most similar to the feature x_i as:

$$j : \underset{d_j}{\text{argmin}} \ |x_i - d_j| \tag{2.17}$$

We match every feature in X with one of D by minimising the distance between the feature x_i and d_j. The labels j of the matching are used to create the histogram as:

$$h(j) = h(j) + 1 \tag{2.18}$$

where h is frequently L_1-normalised after the matching. We create a dictionary from random sampling or clustering algorithms, e.g., *k-means*.

Figure 2.10 depicts an example of two histograms generation. In this example, we generate the Dictionary D via *k-means*. We match the features with the clusters' centroids. The figure depicts the centroids in black. In subsequent sections, we will make use of the BoF algorithm in video analysis. In that case, unlike the example of Figure 2.10, we capture the features from videos.

2.1.9 Fisher Vector

In computer vision, the Fisher Vector (FV) [33] is alternative BoF representation. Instead of characterising an image or video by the number of occurrences of features, we define it by a gradient vector derived from a generative probabilistic model of their features. The gradient of the log-likelihood describes the contribution of the parameters to the generation process. Assuming the set of features $X = \{x_1, \dots x_t, \dots x_T\}$ are generated independently by a mixture model with dis-

tribution parameters θ, we can characterise the set by the following gradient vector:

$$G_\theta(X) = \nabla_\theta \log p\left(x_t|\theta\right). \tag{2.19}$$

To compare two sets of features X and Y, we use a kernel on these gradients, i.e., the Fisher Kernel:

$$\kappa(X,Y) = G_\theta(X)^\top F_\theta^{-1} G_\theta(Y) \tag{2.20}$$

where F_θ is the Fisher Information Matrix:

$$F_\theta = E\left[\left(\nabla_\theta \log p\left(x|\theta\right)\right)\left(\nabla_\theta \log p\left(x|\theta\right)\right)^\top\right] \tag{2.21}$$

Because F_θ is symmetric and positive definite, F_θ^{-1} has a Cholesky decomposition $F_\theta^{-1} = L_\theta^\top L_\theta$, where L is a lower triangular matrix . Therefore $\kappa(X,Y)$ can be rewritten as a dot-product between normalised vectors $\mathcal{F}_\theta$ with: $\mathcal{F}_\theta = L_\theta G_\theta(X)$. Where $\mathcal{F}_\theta(X)$ denotes the FV of the set of features X.

2.1.10 Kernel Function

A kernel is a function that measures the similarity between two feature vectors x_i and x_j in $\mathcal{X}$. To this end, we can represent the similarity via the inner product of the feature space mapping [34]. Let us express the Hilbert space $\mathcal{H}$ [2] via the feature mapping Φ as:

$$\Phi : \mathcal{X} \to \mathcal{H} \tag{2.22}$$

$$x \longmapsto x := \Phi(x), \tag{2.23}$$

The kernel κ is given as:

$$\kappa(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \tag{2.24}$$

The kernel κ measures the similarity of the feature vectors x_i and x_j as the inner product in the Hilbert space $\mathcal{H}$. Figure 2.11 illustrates the Hilbert space mapping. Assume that κ is a real-valued positive definite kernel, and $\mathcal{X}$ is a nonempty set. We define a map from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ into $\mathbb{R}$, denoted as:

$$\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}} \tag{2.25}$$

$$x \longmapsto \kappa(\cdot, x). \tag{2.26}$$

[2] The Hilbert space $\mathcal{H}$ is a complete inner product space. In other words, the product lies in a subset which is also fully enclosed by $\mathcal{H}$.

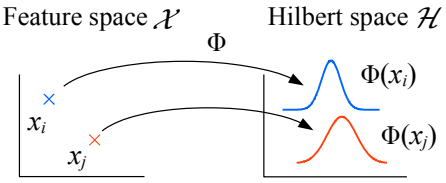


Figure 2.11: Toy example of kernel using a Gaussian function, i.e, $\kappa(x_i, x_j) = \exp\left(- \parallel x_i - x_j \parallel^2 /(2\sigma^2)\right)$.

One can prove that:

$$\langle \kappa(\cdot, x_i), \kappa(\cdot, x_j) \rangle = \kappa(x_i, x_j) \tag{2.27}$$

In future sections, we will use this similarity property to compare two feature vectors via the Fisher kernel. In that case, the feature vectors will represent an entire video.

2.1.11 Fisher Information

Consider the model of distributions p with distribution parameters θ of the observed variable x. The Fisher information measures the overall sensitivity of the functional relationship p to changes of θ by weighting the sensitivity at each potential outcome x with respect to the posterior [35]. The weighting with respect to $p(x|\theta)$ implies that the Fisher information about the parameter θ is an expectation and is given by:

$$F_\theta = E_\theta \left[G_\theta G_\theta^\top \right] \tag{2.28}$$

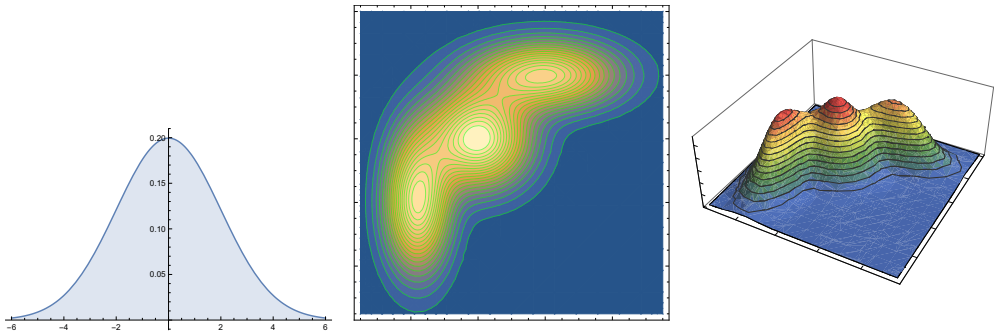
where the score function $G_\theta = \nabla_\theta \log p(x|\theta)$ is the gradient vector of the log likelihood at θ (implicitly depending on x) and the expectation E_θ denotes expectation taken with respect to $p(x|\theta)$. Intuitively, the Fisher information captures the variability of the gradient G_θ.

2.1.12 Gaussian Mixture Model

In probability theory, the Gaussian Mixture Model (GMM) is a joint probability model of Gaussian (more commonly named Normal) distributions [36]. The Normal probability density function, of a continuous real variable x (see Figure 2.12a)), is as follows:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \tag{2.29}$$

where μ and σ^2 represent the mean and variance respectively. For the multivariate case of $x \in \mathbb{R}^D$, the Normal distribution has the form:



a) b) c)

Figure 2.12: a) Normal Distribution probability density function. b) Gaussian Mixture Model contour of a three-component model. c) A surface plot of the overall density of a three-component model.

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x+\mu)\right), \tag{2.30}$$

where Σ is the covariance matrix. The GMM for K Normal distributions is:

$$p(x|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k,\Sigma_k), \tag{2.31}$$

where $\theta = \{\mu_k, \Sigma_k, \pi_k\}$ is the distribution's set of parameters (See Figures 2.12b) and 2.12c)). The parameters are the mean μ_k, covariance matrix Σ_k and weight π_k of the k-th distribution.

The GMM proves to be efficient when no prior knowledge of the data distribution is known. The central limit theorem establishes that the sum of many independent variables approximately has a normal distribution. In video analysis, we do not have prior knowledge of how the video features are initially distributed. Due to this fact, we will make use of the GMM in our models for abnormal event detection.

2.1.13 Expectation Maximisation

We need to estimate the distribution parameters θ of a probabilistic model. One approach is to find a local minimum via a gradient-based algorithm using the negative log-likelihood (NLL) of our set of observed data $\mathcal{D}$ comprising N samples as:

$$\text{NLL}(\theta) = -\frac{1}{N}\log p(\mathcal{D}|\theta) \tag{2.32}$$

However, we have to consider some constraints, such as the covariance must be positive definite, mixing weights must sum to one, etc. that could make the optimisation complicated. To overcome this problem, the Expectation Maximization (EM) [37] is a simple iterative algorithm to find the local optimum. The EM basis is as follows. Let x_i be the visible variables in case i and z_i the missing variables (component labels). The goal is to maximise the log-likelihood of the observed data:

$$\ell(\theta) = \sum_{i=1}^{N} \log p(x_i|\theta) = \sum_{i=1}^{N} \left[\sum_{z_i} p(x_i, z_i|\theta) \right] \tag{2.33}$$

Optimising this expression is hard because we cannot push the log inside the sum. The EM finds a way around this problem by defining the *complete data log likelihood* as follows:

$$\ell_c(\theta) = \sum_{i=1}^{N} \log p(x_i, z_i|\theta) \tag{2.34}$$

This equation cannot be directly computed, so we need to define an expression considering the observed data $\mathcal{D}$ and analyse the parameters θ that maximise the log likelihood in an iterative process. Let us determine the expected complete data log likelihood as:

$$Q(\theta|\theta^{t-1}) = \mathbb{E}\left[\ell_c(\theta)|\mathcal{D}, \theta^{t-1}\right] \tag{2.35}$$

where t is the current iteration number. Q is called the auxiliary function. We consider the expectation with respect to the old parameters θ^{t-1}, and the observed data $\mathcal{D}$. The goal of the expectation step (**E step**) is to compute $Q(\theta|\theta^{t-1})$, or instead, the terms of which the Maximum Likelihood Estimation (MLE) depends on. These terms are known as the Expected Sufficient Statistics (ESS). In the Maximisation step (**M step**) we optimise the Q function with respect to θ:

$$Q(\theta^t) = \underset{\theta}{\text{argmax}} \; Q(\theta, \theta^{t-1}) \tag{2.36}$$

To perform the *maximum a posteriori* estimation the **M step** is as follows:

$$Q(\theta^t) = \underset{\theta}{\text{argmax}} \; Q(\theta, \theta^{t-1}) + log(\theta) \tag{2.37}$$

Expectation Maximisation for GMM: Assuming that the number of mixture components, K, is known, the expected complete data log likelihood is as follows:

$$Q(\theta|\theta^{t-1}) = \mathbb{E}\left[\sum_i \log p(x_i, z_i|\theta)\right] \tag{2.38a}$$

$$= \sum_i \sum_k p(z_i = k|x_i, \theta^{t-1}) \log \pi_k p(x_i|\theta) \tag{2.38b}$$

$$= \sum_i \sum_k \gamma_{ik} \log \pi_k + \sum_i \sum_k \gamma_{ik} \log p(x_i|\theta) \tag{2.38c}$$

where $\gamma_{ik} = p(z_i = k|x_i, \theta^{t-1})$ is the *responsibility* that the component k generates the data z_i from the input x_i. This *responsibility* is computed in the **E step** as follows:

$$\gamma_{ik} = \frac{\pi_k p(x_i|\theta^{t-1})}{\sum_k \pi_k p(x_i|\theta^{t-1})} \tag{2.39}$$

In the **M step**, we optimise Q with respect to θ. Since the expected complete data log-likelihood does not depend on π_k we have:

$$\pi_k = \frac{1}{N} \sum_i \gamma_{ik} = \frac{\gamma_k}{N} \tag{2.40}$$

where $\gamma = \sum_i \gamma_{ik}$ is the weighted number points assigned to component k. To derive the **M step** for the μ_k and Σ_k terms, we analyse the elements of Q that depend on μ_k and Σ_k:

$$\ell(\mu_k, \Sigma_k) = \sum_i \sum_k \gamma_{ik} \log p(x_i|\theta) \tag{2.41a}$$

$$= -\frac{1}{2} \sum_i \gamma_{ik} \left[\log |\Sigma_k| + (x_i - \mu_i)^\top \Sigma_k^{-1} (x_i - \mu_i)\right] \tag{2.41b}$$

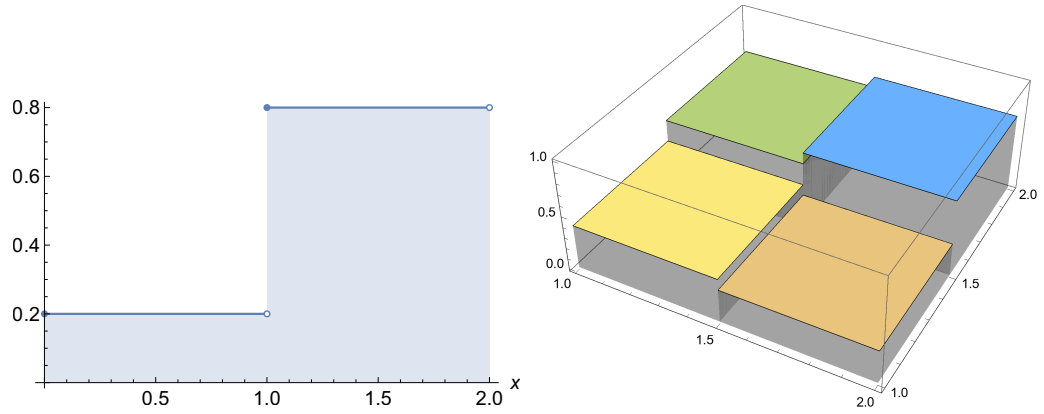
One can show [38] that the new parameters estimates are given by:

$$\mu_k = \frac{\sum_i \gamma_{ik} x_i}{\gamma_k} \tag{2.42a}$$

$$\Sigma_k = \frac{\sum_i \gamma_{ik} (x_i - \mu_i)^\top (x_i - \mu_i)}{\gamma_k} \tag{2.42b}$$

$$= \frac{\sum_i \gamma_{ik} x_i x_i^\top}{\gamma_k} - \mu_k \mu_k^\top \tag{2.42c}$$

After computing the new estimates, we set $\theta^t = \{\pi_k.\mu_k, \Sigma_k\}$ for $k = 1 : K$, and repeat the **E step**. We repeat this process until reaching convergence, in other words, until the parameters do not significantly change across iterations. Another



a) b)

Figure 2.13: a) Bernoulli probability distribution function for one-dimensional variable. b) Bernoulli Mixture Model of a two-component model.

way to stop the process is limiting the number of iterations.

2.1.14 Bernoulli Mixture Model

The Bernoulli Mixture Model (BMM) is a joint probability model of Bernoulli distributions. The Bernoulli distribution is a probabilistic model that has only two possible outcomes [39]. This model has as input the constrained discrete variable $x \in \{0, 1\}$ (See Figure 2.13a)). The probability density function of the Bernoulli distribution is:

$$\mathcal{B}(x|\mu) = \mu^x (1-\mu)^{1-x}, \tag{2.43}$$

where μ is the probability of the variable $x = 1$. For the multivariate case of $x \in \mathbb{N}^D$ the Bernoulli distribution has the form:

$$\mathcal{B}(x|\mu) = \prod_{d=1}^{D} \mathcal{B}\left(x|\mu\right). \tag{2.44}$$

The BMM for K Bernoulli distributions is as follows:

$$p(x|\theta) = \sum_{k=1}^{K} \pi_k \prod_{d=1}^{D} \mathcal{B}\left(x|\mu_k\right), \tag{2.45}$$

where $\theta = \{\mu_k, \pi_k\}$ is the distribution's set of parameters (See Figure 2.13b)). The parameters are the mean μ_k and weight π_k of the k-th distribution.

The BMM is more appropriate to handle binary data [38]. Thus, we use it to model our binary features in subsequent chapters.

Expectation Maximisation for BMM: Following the same procedure described for EM, we can determine the BMM parameters similarly as we explained previously for GMM. In this case, the *responsibility* γ of the feature x_i for k number of components has the same form. Thus, during the **E step** this is calculated as:

$$\gamma_{ik} = \frac{\pi_k p(x_i|\theta^{t-1})}{\sum_k \pi_k p(x_i|\theta^{t-1})}, \tag{2.46}$$

where θ^t is the set of parameters at time t. In the **M step**, Q (Equation 2.38a) is optimised with respect to θ. Since the expected complete data log-likelihood does not depend on π_k we have:

$$\pi_k = \frac{1}{N}\sum_i \gamma_{ik} = \frac{\gamma_k}{N}, \tag{2.47}$$

for N number of x features. To derive the **M step** for μ_k, we analyse the elements of Q that depend on μ_k:

$$\ell(\mu_k) = \sum_i \sum_k \gamma_{ik} \log p(x_i|\theta) \tag{2.48}$$

One can show [38] that the estimate of μ_k is given by:

$$\mu_k = \frac{\sum_i \gamma_{ik} x_i}{\gamma_k} \tag{2.49}$$

After computing the new estimates π_k and μ_k, we set $\theta^t = \{\pi_k, \mu_k\}$ for $k = 1 : K$, and repeat the **E step**. We repeat this process until reaching convergence or a t number of iterations, as in the case of EM for GMM.

2.1.15 Akaike Information Criterion

The Akaike Information Criterion (AIC) is a method to determine the optimal number of components for a mixture model [40]. The approach finds the model components that maximise the likelihood of the observed data with the fewest variables. Given the probabilistic model $p(x_i|\theta)$ with N observations, we define the log-likelihood L for a k number of parameters θ_k as:

$$L_k(\theta) = \sum_{i \in (1,N)} \log p(\mathcal{D}|\theta_k), \tag{2.50}$$

the $\text{AIC}(\theta)_k$ score for k number of parameters is defined as:

$$\text{AIC}(\theta)_k = -2\left(\log\left(L_k(\theta_{\text{MLE}})\right) - \text{dof}(k)\right), \tag{2.51}$$

where $\text{dof}(k)$ is the number of variables included in the model and $L_k(\theta_{\text{MLE}})$ is the Maximised Likelihood Estimate of the posterior for k number of chosen parameters.

The most typical form of Equation 2.51 is for normal distribution. For this model, the maximum log-likelihood for fitting a k-variable model occurs when the standard deviation has the value:

$$\hat{\sigma}_k^2 = \frac{\text{SSE}_k}{N}, \tag{2.52}$$

where SSE_k is the residual sum of squares. So, the maximum value of the log likelihood for this model is:

$$\log\left(L_k(\theta_{MLE})\right) = -\frac{N}{2}\left(\log\left(2\pi\hat{\sigma}_k^2\right) + 1\right), \tag{2.53}$$

and for normal distribution the Akaike score becomes:

$$\text{AIC}(\theta)_k = N\log\left(2\pi\hat{\sigma}_k^2\right) + N + 2k. \tag{2.54}$$

We will make use of Eq. 2.54 in future sections to find the optimal number of components for the mixture models.

2.1.16 K-means

K-means is an algorithm to split data into groups. This algorithm is also a popular variant of the EM algorithm for GMMs [38]. Let us consider a K-component GMM for which we assume that its correlation matrix is $\Sigma_k = 2\sigma I_D$ and its weights $\pi_k = 1/K$ are fixed. Thus, we only have to estimate the cluster centres $\mu_k \in \mathbb{R}^D$. Now consider the following function approximation to the posterior computed during the **E step**:

$$p(z_i = k|x_i, \theta) \approx \mathbb{I}(k = \hat{z}_i) \tag{2.55}$$

where $\hat{z}_i = \text{argmax}_k \; p(z_i = k|x_i, \theta)$ and $\mathbb{I}$ is the mutual information. This expression is sometimes called **hard EM**, since we are making a hard assignment of points to the clusters. Since we assume an equal covariance matrix for each cluster, the most probable cluster $\hat{z}_i$ for x_i can be computed using the ℓ_2 distance, i.e., $\|\cdot\|$ as:

$$\hat{z}_i = \underset{k}{\text{argmin}} \; \|x_i - \mu_k\|_2^2, \tag{2.56}$$

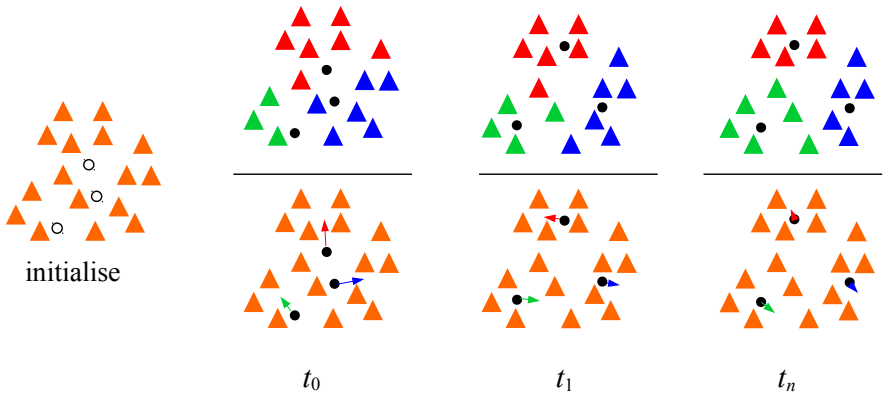


Figure 2.14: K-means algorithm. We initialise the centroids randomly (left). For a number of t iterations or stop criterion, the centroids are calculated (up row) and moved t_n times (down row).

Hence in each **E step**, we must find the distance between N data points and K cluster centres. Figure 2.14 illustrates this process. Given the hard cluster assignments, the **M step** updates each cluster centre by computing the mean of the N_k points assigned to it as follows:

$$\mu_k = \frac{1}{N_k} \sum_{i:z_i=k} x_i \tag{2.57}$$

This clustering process is valid for different distance metrics. We will make use of the k-means algorithm to cluster both double precision data and binary data. For the latter case, we use Hamming distance. In this case, using the xor operation Equation 2.56 becomes.

$$\hat{z}_i = \underset{k}{\operatorname{argmin}} \; x_i \otimes \mu_k, \tag{2.58}$$

whereas equation 2.57 remains the same. We will make use of Eq. 2.58 to cluster our binary data.

2.1.17 Support Vector Machine

Definition

In machine learning, Support Vector Machine (SVM) is a supervised learning model that analyses data for classification and regression purposes [40]. The core idea behind SVMs is to define a boundary between two classes by the maximal separation of the closest points [41].

The underlying intuition of SVMs is the fundamental problem of building a

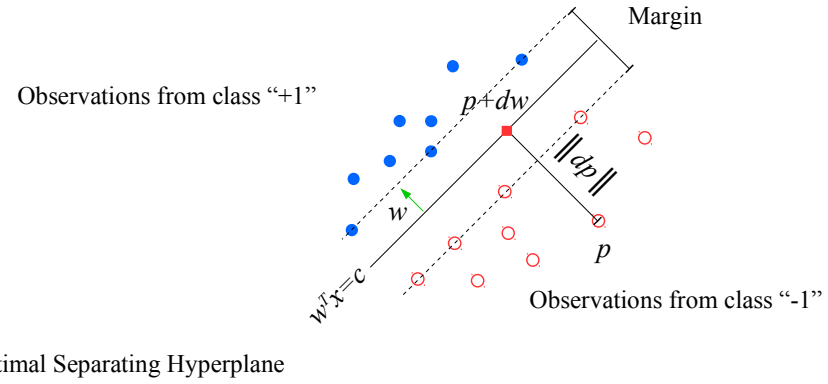


Figure 2.15: The margin is the minimal perpendicular distance between the points in the two data clouds. The optimal separating hyperplane, the solid line, runs down the middle of the margin. The dashed lines run through these points, indicating hyperplanes that are closest to. The point p is d units away from the hyperplane in the direction of w.

classifier for linear-separable data. We can say that a good separating boundary is one that is far from the data. We formalise this concept as a large margin classifier, where the term margin refers to the width of the blank strip separating two data clouds. We define the margin as the shortest perpendicular distance between the hyperplane that separates the data and the observations. Figure 2.15 illustrates the margin definition concept.

Let $w = (w_1, \ldots, w_p)^\top \in \mathbb{R}^p$ be a vector of coefficients and $w_0 \in \mathbb{R}$ be a constant. Given the linear function $\kappa : \mathbb{R}^p \to \mathbb{R}$ as

$$\kappa(x) = w^\top x + w_0 \tag{2.59}$$

We can define an hyperplane as the linear function map that accomplish:

$$\kappa(x) = c, \tag{2.60}$$

The direction vector of a hyperplane is a vector parallel to the hyperplane. A normal vector of a hyperplane is a vector perpendicular to all possible direction vectors of the hyperplane. The SVMs require determining the vector w that maximise the hyperplane separation.

Kernel Machine

We define a kernel machine to be a Generalised Linear Model (GLM), e.g. normal distribution. Given the input feature vector x, its kernelised feature vector is given

as:

$$\phi(x) = [\kappa(x, \mu_1), \ldots, \kappa(x, \mu_K)], \tag{2.61}$$

where μ_k are a set of K centroids of the classes. The principal aspect of kernel machines is to determine the centroids μ_k. To this end, we select a loss function L as the ℓ_2-regularised empirical risk function J:

$$J(w, \lambda) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \lambda \parallel w \parallel^2, \tag{2.62}$$

where y_i is the class label, $\hat{y}_i = w^\top x_i + w_0$ and λ is a constant factor. Recalling that $x \in \mathbb{R}^D$, for N number of feature vectors we define X as the corresponding $N \times D$ design matrix. The problem can be established in matrix notation as:

$$J(w, \lambda) = (y - Xw)^\top (y - Xw) + \lambda \parallel w \parallel^2, \tag{2.63}$$

where the optimal solution is given by:

$$w = (X^\top X + \lambda I_D)^{-1} X^\top \tag{2.64a}$$

$$y = \left(\sum_i x_i x_i^\top + \lambda I_D \right)^{-1} X^\top y. \tag{2.64b}$$

We can rewrite these equations in the form of inner products. To this end, we use calls to a kernel function κ. This form is kernelised, but not sparse. However, if we replace the loss L with some other loss function, we can ensure that the solution is sparse so that predictions only depend on a subset of the training data, known as support vectors.

The combination of the kernel plus a modified loss function is the SVMs basis. This technique was originally designed for binary classification but extrapolates regression and multi-class classification.

SVMs for binary classification

To classify data observations with two possible labels $y \in \{-1, 1\}$, it is necessary to determine the loss produced by choosing them. The loss has the form:

$$L(y, \eta) = \max(0, 1 - y\eta) = (1 - y\eta)_+ \tag{2.65}$$

where

$$\eta = f(x) = w^\top x + w_0 \tag{2.66}$$

is the log odds ratio. Here $\eta = f(x)$ is our "confidence" in choosing label $y = 1$; however, it should not have any probabilistic semantics. The overall objective is given as:

$$\min_{w,w_0} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{N} (1 - y_i f(x_i))_+, \tag{2.67}$$

where C is the regulariser factor. This expression is non-differentiable. However, by introducing slack variables ξ_i , one can show that this is equivalent to solving:

$$\min_{w,w_0,\xi} \frac{1}{2} \parallel w \parallel^2 + C \sum_{i=1}^{N} \xi_i \;\; s.t. \;\; \xi_i \geq 0, y_i(x_i^\top + w_0) \geq 1 - \xi_i, i = 1 : N \tag{2.68}$$

for N number of $x \in \mathbb{R}^D$ vectors. This expression is a quadratic program in $N+D+1$ variables, subject to $O(N)$ constraints. We can eliminate the primal variables w, w_0 and ξ_i, and just solve N dual variables, which correspond to the Lagrange multipliers for the constraints. Standard solvers take $O(N^3)$ time. One can show that the solution has the form:

$$\hat{w} = \sum_i \alpha_i x_i \tag{2.69}$$

We name the x_i for which $\alpha_i > 0$ are support vectors; these are points which are either incorrectly classified, or are classified correctly but are on or inside the margin. At test time, the prediction is made using the decision function:

$$\hat{y}(x) = \text{sgn}\,(f(x)) = \text{sgn}(\hat{w}_0 + \hat{w}^\top x) \tag{2.70}$$

Using Equation 2.69 we have:

$$\hat{y}(x) = \text{sgn}\left(\hat{w}_0 + \sum_{i=1}^{N} \alpha_i \kappa(x_i, x)\right) \tag{2.71}$$

This evaluation takes $O(sD)$ time to compute, where $s \leq N$ is the number of support vectors. The time depends on the sparsity level, and hence on the regulariser C.

SVMs for multi-class classification

Upgrading an SVM to the multi-class case is elaborated since the outputs are not on a calibrated scale and hence are hard to compare to each other [38]. The naive approach is to use a one-versus-the-rest approach (also called one-vs-all), in which we train K binary classifiers, $f_k(x)$, treating the data from class k as positive and the data from all the other classes k' as negative. Another approach is to use the

one-versus-one or OVO approach, also called all pairs, in which we train $K(K-1)/2$ classifiers to discriminate all pairs $f_{(k,k')}(x)$. We then classify a point into the class which has the highest number of votes.

2.1.18 Nigstrom Approximation Kernel Extreme Learning Machine

The Nigstrom Approximation Kernel Extreme Learning Machine (nAkELM) is single-hidden layer feed forward neural network [42]. Let us consider a set of vectors $x_i \in \mathbb{R}^D$ such as $i \in [1, \ldots, N]$ with associated classes $c_i \in \{1, \ldots, N\}$ to train the neural network. The network consists of D inputs, L hidden and C output neurons. The elements of the network target vectors $t_i = [t_{i1}, \ldots, t_{iC}]^T$, each corresponding to a training vector x_i , are set to $t_{ik} = 1$ for vectors belonging to class k, i.e., when $c_i = k$, and to $t_{ik} = -1$ when $c_i \neq k$.

In nAkELM-like approaches, the network input weights $W \in \mathbb{R}^{D \times L}$ and the hidden layer bias values $b \in \mathbb{R}^L$ are randomly assigned, while the network output weights $W \in \mathbb{R}^{L \times C}$ are analytically calculated by storing the network hidden layer outputs $\phi \in \mathbb{R}^{D \times L}$ corresponding to all the training vectors $x_i : i \in [1, \ldots, N]$ in $\phi = [\phi_1, \ldots, \phi_N]$, as $O = W_{out}^T \Phi$, where $O \in \mathbb{R}^{C \times N}$ is a matrix containing the network responses for all training data x_i. The problem of mapping the vectors x_i is solved by minimizing the W_{out} error projection problem:

$$\mathcal{J} = \frac{1}{2}\left\|W_{out}\right\|^2 + \frac{\lambda}{2} \sum_{1 \leqslant i \leqslant N} \|\xi_i\|_2^2, \tag{2.72}$$

where $\xi_i \in \mathbb{R}^C$ is the error vector corresponding to x_i and $\lambda > 0$ is a parameter denoting the importance of the training error in the optimisation problem, which is subject to:

$$W_{out}^\top \phi_i = t_i - \xi_i. \tag{2.73}$$

By solving this problem, the network optimises the x_i vectors mapping to the class labels t_i. We can use then $W_{out}^\top$ to determine the class of new observed vectors.

2.1.19 Markov Models

Definition

In probability theory, a Markov model is a stochastic model used to model randomly changing systems. The central principle behind a Markov model is to assume that input variable x poses all the relevant information for predicting future observations [38]. If we assume T discrete time steps, we can write the joint distribution for the

Markov Model as follows:

$$p(x_{1:T}) = p(x_1)p(x_2|x_1)p(x_3|x_2)\ldots = p(x_1)\prod_{t=2}^{T} p(x_t|x_{t-1}) \tag{2.74}$$

This expression is called a Markov chain or **Markov model**. If we assume the transition function $p(x_t|x_{t-1})$ is independent of time, then the chain is called homogeneous, stationary, or **time-invariant**. This assumption is an example of parameter tying, since we share the same parameters by multiple variables. This assumption allows us to model an arbitrary number of variables using a fixed number of parameters; such models are called **stochastic processes**. If we assume that the observed variables are discrete, so $x_t \in \{1, \ldots, K\}$, this is called a discrete-state or Finite-State Markov Chain (FSMC).

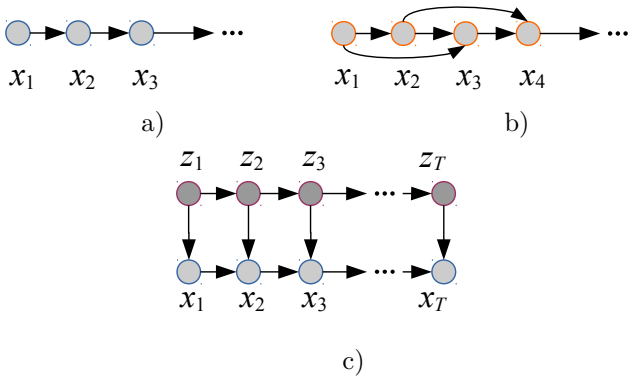


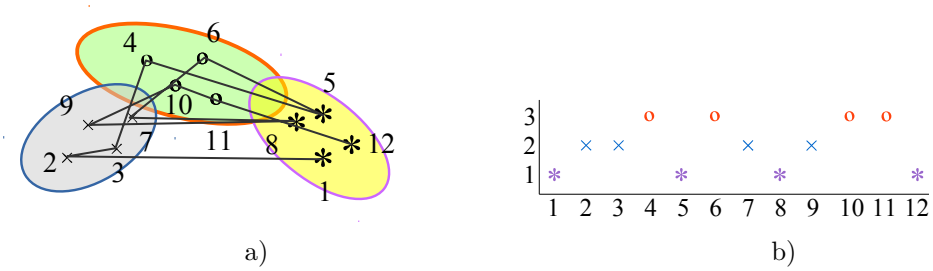
Figure 2.16: a) First and b) second-order Markov chain. c) A first-order Hidden Markov Model.

Figure 2.16a) illustrates a first-order Markov chain as a Directed Acyclic Graph (DAG). We can assume that the immediate past, x_{t-1}, captures all we need to know about the entire history, $x_{1:t-2}$. We can relax the assumption a little by adding a dependence from x_{t-2} to x_t; this is called a **second-order Markov chain**, Figure 2.16b) illustrates this model. The corresponding joint Markov model has the following form:

$$p(x_{1:T}) = p(x_1, x_2)p(x_3|x_1, x_2)p(x_4|x_2, x_3)\ldots \tag{2.75}$$

whose product form is:

$$p(x_{1:T}) = p(x_1, x_2)\prod_{t=3}^{T} p(x_t|x_{t-1}, x_{t-2}) \tag{2.76}$$



a) b)

Figure 2.17: a) 2D data example from a 3-state HMM. Each state emits from a 2D Gaussian. b) The hidden state sequence.

We can create higher-order Markov models in a similar way. Unfortunately, even the second-order Markov assumption may be inadequate if there are long-range correlations amongst the observations. We cannot keep building ever higher order models, because the number of parameters significantly increases. We see the solution in the next subsection.

Hidden Markov Model

An alternative approach to control the number of parameters of a Markov model, is to assume that there is an underlying hidden process and we can model it by a first-order Markov chain. The result is known as a Hidden Markov Model (HMM), Figure 2.16c) illustrates this model. For this example, z_t is known as a hidden variable at time t, and x_t is the observed variable. HMM consists of a discrete-state Markov chain, with hidden states $z_t \in \{1, \ldots, K\}$, plus an observation model $p(x_t|z_t)$ given in Eq. 2.76 .

$$p(z_{1:T}, x_{1:T}) = p(z_{1:T})p(x_{1:T}|z_{1:T}) = \left[p(z_1)\prod_{t=2}^{T} p(z_t|p(z_{t-1})\right]\left[\prod_{t=1}^{T} p(x_t|z_t)\right] \quad (2.77)$$

The observations in an HMM can be discrete or continuous. If they are discrete, it is common for the observation model to be an observation matrix:

$$p(x_t = l|z_t = k, \theta) = B(k, l), \quad (2.78)$$

where B is the beta function, l and k are transition states. If the observations are continuous, it is common for the observation model to be a conditional Gaussian. Given a Gaussian model with parameters $\theta = \{\mu_k, \Sigma_k\}$ we have:

$$p(x_t|z_t = k, \theta) = \mathcal{N}(x_t|\mu_k, \Sigma_k) \quad (2.79)$$

Figure 2.17a) shows an example where we have 3 states, each of which emits a different Gaussian. The resulting model is similar to a Gaussian mixture model, except that the cluster membership has Markovian dynamics. Figure 2.17b) shows the hidden transition sequence of this example.

Transition Matrix

When the input x_t of the HMM is discrete, we can write the conditional distribution $p(x_t|x_{t-1})$ as a squared matrix, known as the **transition matrix** A, where $A_{ij} = p(x_t = j|x_{t-1} = i)$ is the probability of going from state i to state j in one step. Each row of the matrix sums to one, $\sum_j A_{ij} = 1$. We define the n-step transition matrix $A(n)$ as:

$$A_{ij}(n) = p(x_{t+n} = j|x_t = i), \tag{2.80}$$

which is the probability of getting from i to j in exactly n steps. One advantage of the transition matrix, is that we can evaluate the FSMC probability transitions in $O(n)$ time via a single memory reading. Thus, this matrix is highly efficient regarding speed.

2.2 Abnormal Event Detection

Within this computer vision field, we find two main approaches: accuracy first (Section 2.2.1) and speed first (Section 2.2.2). The former focuses on detecting unusual events with high accuracy no matter the required processing time. The latter explicitly aims at practical applications for video surveillance. Hence, a short processing time is a priority for these approaches.

For both approaches, it is necessary to process STSRs because these are the one to be analysed. Therefore, we first describe mechanisms to define the STSRs. An array of sorted cells comprises the STSRs, where each cell has the location information of a particular spatio-temporal region to capture the motion source. Thus, the cell structure [3] contains the STSRs locations from where we scan the sequences. To this end, there are mainly four strategies to extract features.

1. No-overlapped[4] single-scale scanning: most approaches extract features using only one spatial scale of size $n_x \times n_y$[5] for the whole frame, without overlapping

[3] The proposed cell structure contains the spatio-temporal location to encode the video, but we also use it for local composition representations in subsequent sections.

[4] We refer to the overlap as two spatio-temporal regions whose spatial intersection is not the empty set $\emptyset$.

[5] The size $n_x \times n_y$ is the spatial size of the sampling window.

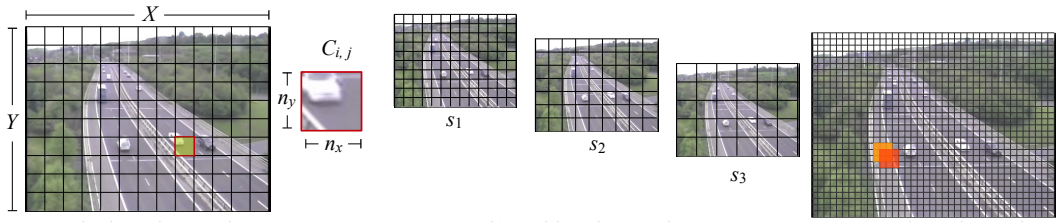


Figure 2.18: a) Single scale scanning with a support region of size $n_x \times n_y$, each support region located at position (i, j) is called cell $c_(i, j)$. b) Multi-scale scanning using three different scales. c) Overlapped scanning, two support regions (yellow and orange) overlap during the scanning.

as in [43–45]. See Figure 2.18a for an example.

2. No-overlapped multi-scale scanning no overlapped: the scanning process takes place at many scales s, i.e. $n_{x,s} \times n_{y,s}$, without overlapping as in [46, 47], See Figure 2.18b for an example.

3. Overlapped single-scale scanning: the frame is processed at one scale, but the STSRs share pixels among them as in [48, 49]. See Figure 2.18c for an example.

4. Overlapped multi-scale scanning: many scales and overlaps are present in this technique as in [4, 50].

The number of samples and number of scales selected is also known as density. Denser methods achieve better performance as in [4, 50, 51]. However, we have to highlight that the complexity of the dense scanning methods is very high.

The work of Bertini et al. [50] shows that extracting features from overlapped multi-scale STSRs (dense scanning) enhances detection precision, but severely increases the computational time. In their paper, they report the accuracy in terms of the overlapping region vs FPS, varying from 10 FPS to 80 FPS with an overlap percentage from 10% to 60%. The computational time is eight times longer when the overlap in the STSRs doubles. The authors notice that the amount of extracted features of the no-overlapped single -scale algorithm has little influence on the accuracy of the method. But in the case of the overlapped multi-scale scanning, the amount of extracted features affects the precision of the approach and strongly increases the computational time. Based on their work, one can see that the vast number of scales it is recommended because some objects in the scene may look bigger than others due to their position. In order words, small objects may require small sup-

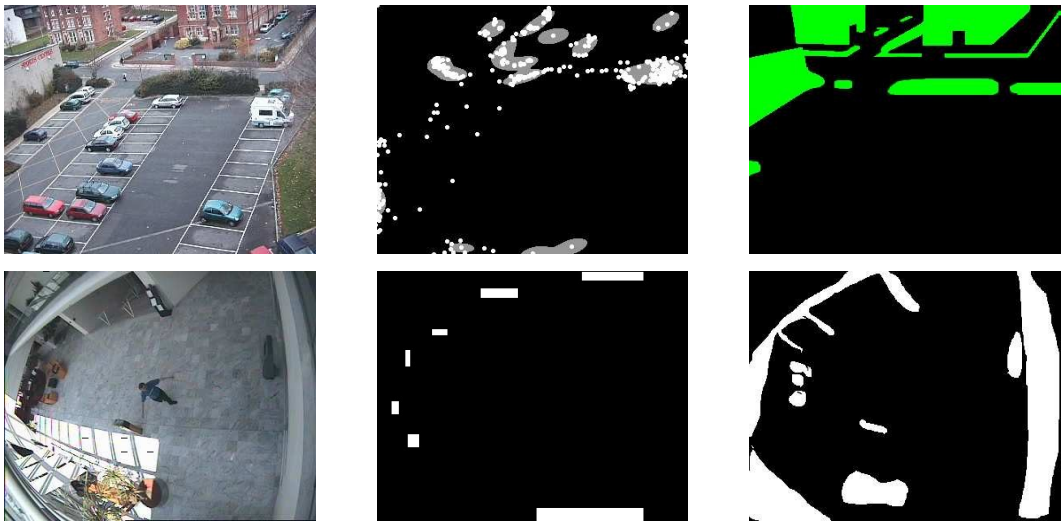


Figure 2.19: Illustrations from [52]. First column, scenes from the parking lot and indoor sequences. Second column, exits/entrances of the scenes. Third column, obstacles in the scenes.

port regions while large objects may need large regions. Therefore we conclude that multi-scale scanning is computational costly as the overlapping vs FPS experiment confirmed. The dense-scanning method of Bertini et al. [50] generate thousands of features for the video sequences. Another highly accurate method proposed by Roshtkhari and Levine [4] produces even millions of features. These features are generated by an overlapped multi-scale scanning technique applied to each pixel of the video. Consequently, there is a trade-off between detection accuracy and the number of extracted features.

2.2.1 Accuracy First Scope

Under this scope, we can find two main categories: trajectory-based and no-trajectory-based abnormal event detection.

Trajectory-based Abnormal Event Detection

This computer vision field started with the work of Dee and Hogg [52] in 2004 evaluating pedestrian movement. The objective was to determine which trajectories in the sequences are *explicable* considering the previously seen ones. To this end, they employ a semi-parametric machine state model. The machine evaluates the trajectory cost through the states by considering that *normal* behaviour should take place when the pedestrians visit specific areas and avoid the obstacles in the

scene. Each area and obstacle in the scene has a label. Thus, by going between two points in the scene the model should follow a sequence of labels which *explains* the movements. The model captures this information in the form of transitions. Their experiments were carried out with two scenes (see Figure 2.19). The evaluation consists of determining which trajectories are of the operator's interest, ranking 0 if they are irrelevant and 5 if they are the most relevant. They discovered that there is a significant relationship between the cost obtained by the model and the operator's interest. The importance of this work is that it reveals the possibility to explain unusual behaviour mathematically. This idea was reinforced by Makris and Ellis [53] automatically detecting the exit/entrance zones and capturing the transitions with an HMM.

Detecting abnormal trajectories is improved by the work of Porikli and Haga [54], which incorporated visual descriptors to identify unusual events. They explore a series of features obtained from tracking objects from the scene, e.g. size, speed, colour, etc., and proceed to analyse the likelihood of appearance via an HMM. Clustering the trajectories prior model generation demonstrates advantages as Piciarelli and Foresti [55], Piciarelli et al. [56] show. Using a tree of clustered trajectories is an efficient way to detect unusual events. Piciarelli et al. [56] populate the tree with probabilities of the seen trajectories in specific areas of the scene.

Tracking multiple motion sources, e.g. colour, texture to detect abnormal events also is explored by Li et al. [57]. Tracking is improved using SVM by Ivanov et al. [58] and by Zhang et al. [59]. In the work of Zhang et al. [59], they classify significantly better the trajectories compared with the methods previously used by [57]. Jiang et al. [60] proposed tracking incorporating spatio-temporal context. In this work, they detect abnormal trajectories in two inputs spaces, i.e., colour and texture.

Calderara et al. [61] explore tracking models by employing graphs to detect abnormal events. They describe the trajectories as paths through the graph's vertices. Tung et al. [62] proposed trajectory goals. They suggest analysing not only the trajectories but a collection of them to find specific behaviour. Mo et al. [63, 64] explore sparse reconstructions for abnormal trajectory detection. Their proposed method has been proven to be robust to occlusions associated with sparse/weak or occluded trajectories.

No Trajectory-based Abnormal Event Detection

In this category, we find the abnormal event detection techniques formulated without requiring objects tracking. The work of Zhong et al. [65] introduces the concept of

video words for abnormal event detection. They propose event representation via histograms where the words are video volumes.

Au et al. [66] propose extracting features to represent scenes. They employ similarity measures, e.g. Euclidean and Minkowski distances, between frames to detect the abnormal ones. The detection is done by storing a considerably big set of *normal* frames and later comparing all the stored frames with the new observed frames. They introduce this concept specifically for video surveillance.

Andrade et al. [67] introduce the usage of optical flow to detect abnormal events. From small STSRs, they extract optical flow and analyse it via an HMM. The nth component of a GMM gives the transitions between the HMM for the observed optical flow. Archetti et al. [68] retake this concept by using small STSRs in the scene capturing the optical flow and by individually modelling each region. Pruteanu-Malinici and Carin [69] improve previous Markov models by infinite Hidden Markov Models (iHMM). This improvement allows the system to analyse longer sequences.

Ali and Shah [70] employ optical flow in extremely crowded scenes; they introduce an optical flow segmentation strategy to reveal common patterns in such scenes. The method can detect unusual patterns as a result of unusual behaviour. This behaviour could be for instance a stampede, where the abnormal event is given not by isolated individuals but the crowd. In the same light, instead of segmentation, Mehran et al. [71] explore to join and analyse several random sampled regions in sequence to detect unusual optical flow. Crowded scenes are also explored by Kratz and Nishino [72] using an HMM with competitive results.

Evaluating the influence of the camera's position when extracting features is presented by Wang et al. [73]. Abnormal detection methods attain higher accuracy when they compensate for the apparent distortion generated by the camera's position.

Finding unusual patterns that signal the presence of an abnormal event in the context of video words compositions is explored by Boiman and Irani [10]. In their work, they consider irregular compositions of spatio-temporal regions to detect unusual behaviour. Each video volume has a unique probability of occurring in space and time. The model analyses the joint probability of several video volumes in specific space and time locations.

Analysing spatio-temporal compositions is also addressed by Benezeth et al. [74]. In their work, they propose to capture information from word co-occurrences. The assumption is that some words co-occur with other words with higher probability. Detecting abnormal events using Gaussian Regression Process (GRP) is explored by Loy et al. [75]. GRP reveals advantages over HMM because the number

of transition states, connectivity and typology represent a complex problem to be estimated.

Kim and Grauman [76] put graph theory to the test. They propose to model the video sequence as a connected graph using Markov Random Field (MRF), associating the spatio-temporal location with nodes and the vertices with the transitions. At each vertex, they compute optical flow features and analyse their composition using the MRF. The video is also modelled as a fully connected graph by Thida et al. [45]. They explore the reconstruction cost function of the Laplacian associated to the graph.

Zaharescu and Wildes [44] explore long-term histograms for individual isolated regions of the scene. Their method proves to be computationally efficient because features are extracted only from areas activated by a motion threshold. Mahadevan et al. [77] explore Mixture of Dynamic Textures (MDT) to replace traditional GMM. The model incorporates the time domain to the posterior likelihood. Thus, unlike GMM-like models, the model can capture the probability of a particular texture to occur at a specific time. Li et al. [78] improve MDT by adding a saliency detector in the space domain to enhance detection.

Feng et al. [79] explore Self Organising Maps (SOM) to detect abnormal events. Their method can enhance accuracy detection by updating the system with the newly observed features.

Yubing et al. [80] explore foreground as motion source to detect abnormal events. This new motion source provides the system capability to detect unusual long-term events, e.g. loitering and abandoned luggage. Capturing the foreground is also explored by Reddy et al. [81]. Individual cells are employed to detect large objects and long-term events. Their approach is very computational efficient because they only process those STSR whose foreground is above a threshold. Thus, they do not extract features from a significant number of STSRs. One significant advantage of their work is that they can detect abnormal long-term events.

Automatically scene segmentation to analyse each region individually is proposed by Loy et al. [82]. The segmentation represents moving parts associated with trajectories of the same direction. Guo et al. [83] improve foreground features by applying Principal Component Analysis (PCA) prior computate the model.

Zhao et al. [84] are the first to explore sparse reconstructions. The cost of the feature reconstruction given by dictionaries is employed as a criterion to detect abnormal events. Sparse reconstruction using optical flow histogram is later proposed by Cong et al. [85, 86, 87] with significantly better results than [84]. Saligrama and Chen [88] describe sparse reconstructions as a particular case of k-Nearest Neigh-

bour (kNN). Using background for sparse reconstruction is proposed by Tran et al. [89], and proved to be a reliable motion source. In the context of sparse representation, many other features successfully attain excellent performance. These include wavelets features by Zhu et al. [90], 3DHOF by Zhu et al. [91], trackless features by Mousavi et al. [92] and holistic features by Marsden et al. [93].

Bertini et al. [50, 51] explore multi-scale and overlapped feature extraction, Xu et al. [94], Feng et al. [95] explore hierarchical feature extraction, and [4, 96, 97] investigate dense feature extraction. All those proposed feature extraction methods have been shown to be more efficient than single scale extraction. Extracting features in multiple scales proved to be efficient as Roshtkhari and Levine [4] report. They use as motion information source the spatio-temporal location of a large number of time-gradient features. They merge all those features and their spatio-temporal locations in a single vote observation model for each pixel of the scene. Unfortunately, all these methods are very time-consuming and thus impractical for real scenarios.

Cheng et al. [98, 99] propose encoding STSRs detected by STIPs. Modelling STIPs has been shown to be efficient because the identified regions are very descriptive locations that capture the dynamics of the scene. High order mapping of the STIPs detected is proposed by Zhao et al. [100] with competitive accuracy.

2.2.2 Speed First Scope

Adam et al. [1] are the authors that firstly explore this more recent category of abnormal event detection. They suggest that abnormal event detection methods should be designed considering large-scale operation needs. In their work, they propose to analyse the optical flow for individual regions of the scene, i.e. Local Optical Flow (LOF). To this end, they divide the frame into sub-windows (they call them monitors) and record the optical flow (see Figure 2.20). They generate an ad-hoc exponential function to describe the observed optical flow. Future observations are classified as abnormal/normal by thresholding the ad-hoc function. They implement the method in a Digital Signal Processing (DSP) architecture achieving online processing times[6] which demonstrate remarkably speed efficiency. They test their approach in public scenarios where people walking is deemed to be normal; therefore one possible abnormal event is people running. They also examine people's behaviour at a subway exit. In that case, we expect people walking only in one direction. Their method detects people running and walking in opposite directions.

[6] Although this method meets the required processing times for practical video surveillance, it is not specified some of the static video format characteristics, i.e. frame size, frame rate. Therefore it is not possible to determine for which video format the proposed method achieves online processing times.

Figure 2.20: Illustrations from [1]. (left) A typical scene and the positions of the multiple low-level monitors chosen for this scene. (right) The long-term average optical flow magnitude at each monitor of the scene. Note that monitors closer to the camera indeed observe larger optical flow.

The importance of their work is that it reveals that unusual events can be detected immediately as they occur in the scene, thus warning the operators in real time[7] of what has just happened. The proposed method, unfortunately, has three main flaws listed as follows:

1. It only detects the events that are depicted **close to the camera**.
2. It cannot detect **long-term abnormal events** (e.g., loitering).
3. It cannot detect **unusual compositions** (e.g. fighting).

The events of the last point are those in which the optical flow might not be abnormal, but the composition of the cells that capture it might be. In recent years, other authors have focused on solving the composition problem.

Lu et al. [47] propose a fast sparse reconstruction method. They suggest speed up computations by extracting temporal differences at different scales as features and proceed to evaluate the sparse cost reconstruction of a feature dictionary, i.e. BoF. The method's implementation can process frames up to 150 FPS. One of the key aspects to achieving such processing times is that extracted features are flattered vectors of the temporal gradients; thus no orientation or polynomial algorithms (e.g. as required by optical flow) are needed to gather those features. Another important aspect is that the sparse reconstruction takes place at different but few large scales, so the total number of features processed is relatively small.

[7] In the context of real time systems, the video frame rate is considered the hard limit for the system's response. Under this light, any system must give the operators the response within the hard limit to ensure that the response time never generates delays. Thus, online systems are real-time systems for which the hard limit is determined by the FPS.

Biswas and Babu [46], Biswas and Venkatesh Babu [101] propose extracting fast features using the compressed motion vectors, e.g. H264, as feature vectors. Thus, the video compressed itself generates the features. They detect abnormal events by evaluating the sparse representation cost of the motion vectors to a given dictionary. In their work, they select features at various scales to enhance performance.

2.2.3 Speed First vs Accuracy First

Accuracy first methods (Section 2.2.1) have seen important contributions over the past decade. They usually attain good performance at the expense of increasing frame processing times. One common characteristic of these methods is the technique to select the spatio-temporal regions of the scene to be modelled and analysed. Such technique could be dense scanning [4, 97], multi-scale scanning [50, 85, 102] and convolution-based STIP detection [98, 99]. These techniques usually provide sufficient data to capture the scene's dynamics and spatio-temporal compositions; however, the number of spatio-temporal regions selected for the analysis may result in a large number of features to be processed [4, 50, 85, 88, 97–99]. Although important efforts have been made to reduce the complexity associated with the definition of a scene's spatio-temporal compositions [4, 98], many of the proposed improvements may still require considerably long computations [4, 97, 98].

Another essential characteristic of these *accuracy first* methods is their high descriptive features used to improve performance. Among these, optical flow features increase detection accuracy [90, 91]. For example, in [90] the authors propose an entirely unsupervised non-negative sparse coding-based approach that employs HOF to identify abnormalities in crowded scenes with promising performances. In [85], the authors adopt Multi-scale HOF, which preserves temporal contextual information, to detect anomalies in crowded scenes. Computing such descriptive features, unfortunately, require long processing times [4, 90, 92, 97, 102]. For example, local descriptors computed using dense scanning techniques have been shown to improve performance, but at the expense of repeated computations [4, 50].

Speed first methods (Section 2.2.2) have recently gained interest within the area of video anomaly detection [4, 46, 47]. These methods usually reduce computational times by reducing the number of features to be processed per frame [46, 47, 81] or employing local low-complexity descriptors [4, 46, 47, 98].

We stress positive aspects of speed first methods. The work of Lu et al. [47] and that of Biswas and Babu [46] manage to model a small number of features even though they employ multi-scale scanning techniques. *Speed first* methods may also

use features that are fast to compute, but not highly descriptive. For example, in [46] the authors employ the motion vectors of a video sequence as features in a histogram-binning scheme. In [47], the authors use local temporal gradients extracted in a multi-scale fashion as the main features.

Another conventional technique to reduce the processing times is by employing cell-based methods to extract features from fixed spatio-temporal regions [50, 81, 85, 92]. Cell-based approaches, therefore, do not require STIP or other saliency detection techniques; moreover, they can be used to limit the number of extracted features [50].

2.3 Spatio-Temporal Features

At the core of video analysis are the spatio-temporal descriptors. We require these for abnormal event detection and action recognition. However, very common authors test new proposed features only on action recognition. Broadly speaking, we can classify them into two main sub-categories. Double precision features (Section 2.3.1) and Binary features (Section 2.3.2) according to the data type representation. We describe them next.

2.3.1 Double Precision Features

Extracting features with the specific purpose of video analysis probably started with the work of Dollar et al. [2]. Their goal is to classify the mouse behaviour out of five actions: drink, eat, explore, groom and sleep (see Figure 2.21). Although a few approaches precede their work to identify actions in the video; they introduce the concept of spatio-temporal locations being mapped into a different feature space to boost their descriptiveness. In detail, their work follows the idea of Laptev and Lindeberg [103] that particular spatio-temporal locations in the video are more distinctive than others to identify actions. They evaluate a series of metrics to rank the descriptiveness of those regions. These metrics are normalised flattered pixel values of brightness, gradients and optical flow (Lucas-Kanade). Surprisingly, the best performing metric is brightness gradients. The worst is normalised pixel values. The difference between the best and the worst metrics is around 20% classification error. This disparity reveals the importance of raw data transformation. This work also exhibits that 2D descriptors (e.g. SIFT) have inferior performance in action recognition. Therefore, it is important to include temporal information in the features. It is still not clear why the optical flow did not perform the best. Perhaps,

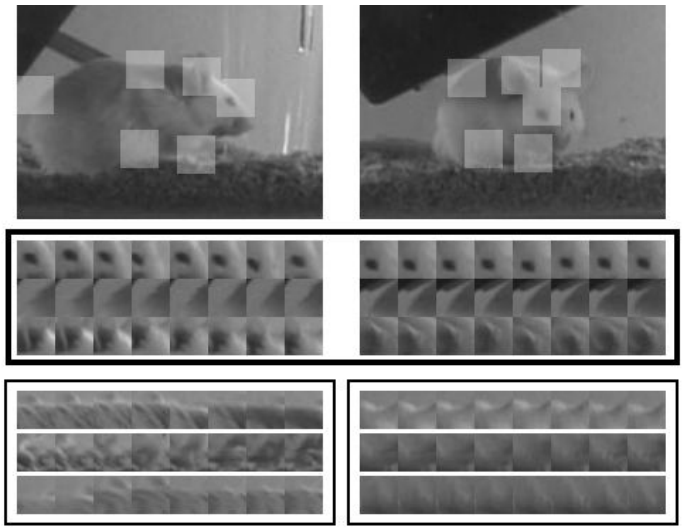


Figure 2.21: Illustrations from [2]. First row, Spatio-temporal support regions detected by interest points. Second and third rows, patches contained in the video volume.

it is the fact that the tested videos contained relatively simple background[8]. Future work reveals that optical flow features perform better in action recognition [6, 7].

Scovanner et al. [11] propose an extension of the popular 2D SIFT [104] descriptor for the spatio-temporal domain. 3DSIFT employs as motion source spatio-temporal gradients, making the descriptor robust to intensity changes. They generate the feature vector by binning the gradients' direction using the polar and azimuthal angle of all pixels enclosed in the support region. The binning mechanism has been shown to be efficient with 32 orientations. They use a 3DSIFT + BoF +SVM pipeline to classify ten actions achieving an accuracy higher than 80%.

Extending 2D feature basis to the temporal domain is also proposed by Willems et al. [105], i.e., 3D extended Speeded Up Robust Features (eSURF). The descriptor captures the Haar-wavelets responses of the spatio-temporal region convoluted with 3 differential operators. This descriptor is not as accurate as 3DSIFT but is considerably faster.

Klaser et al. [6] propose to capture spatio-temporal gradients using high order histogram binning. Similar to HOG, they propose 3DHOG, which is a feature vector with less than a half the dimensions of 3DSIFT. Thus it requires less computational effort for its calculation.

Laptev et al. [106] propose an orientation binning scheme using the optical

[8]Henceforward, we refer to simple background to those scenes which contain no significant camera motion and environmental conditions that could easily allow segmenting moving objects.

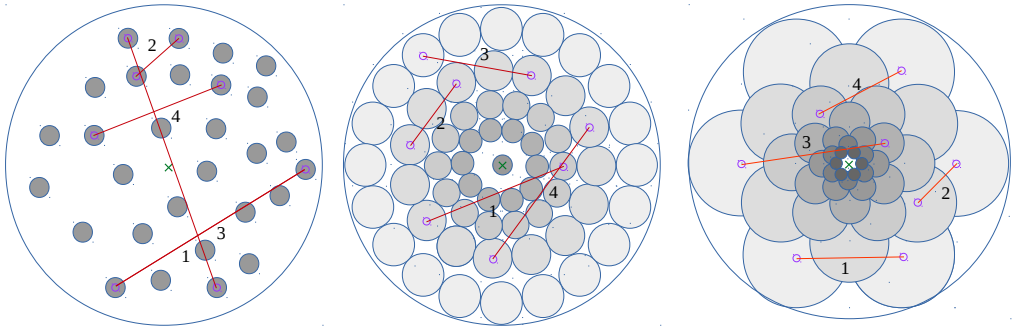


Figure 2.22: (left) BRIEF-ORB patterns, (centre) BRISK and (right) FREAK. Four regions are compared around the interest point (green cross) to create logical values.

flow as motion source, i.e. HOF. Their proposed feature has been shown to be very efficient because of its descriptiveness and considerably lower dimensionality. Their less than a hundred-dimensional vector is very accurate in complex backgrounds outperforming gradient-based motion sources, such as HOG. In [107], optical flow has been shown to be more efficient than spatio-temporal gradient using different STIP detectors. Extracting derivatives from the optical flow before binning demonstrated to be descriptive by Wang et al. [7].

Wang et al. [7, 73, 108] demonstrate that tracking trajectories to create features from dense optical flow instead of STIP detectors significantly improves accuracy in action recognition. These approaches capture the small vector displacements of the optical flow to create a trajectory.

Oliva and Torralba [109], Shao et al. [110], Shao and Gao [111] explore frequency transformations to generate features capturing the main frequency components of the video volume. Oriented filters in the frequency domain, as proposed by Solmaz et al. [112] demonstrate competitive accuracy as well.

2.3.2 Binary Features

Image Features

The authors of [113–116] extensively explore binary descriptors for images. Calonder et al. [113] propose Binary Robust Independent Elementary Features (BRIEF), which is a descriptor that encodes into binary strings the support region around an interest point. To this end, a random pattern is use to compare different regions (Figure 2.22 left). Rublee et al. [114] propose Oriented fast and Rotated BRIEF (ORB), which is a descriptor that uses random patterns as BRIEF However, it estimates the orientation of the interest point [117] before the encoding. Leuteneg-

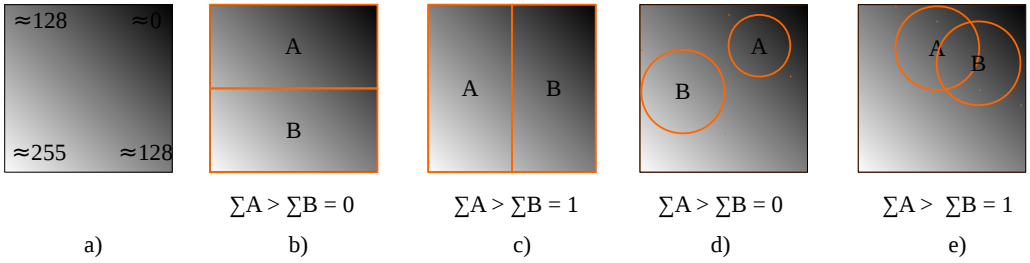


Figure 2.23: A toy patch a). Viola-Jones patterns (b-c), FREAK patterns (d-e) generating logical values. This generation is done by comparing the sum of pixel's intensities in the regions A and B.

ger et al. [115] propose Binary Robust Independent Elementary Features (BRIEF), which is a descriptor that compares regions of variable size in a circular pattern enclosing the interest point (Figure 2.22 centre). Alahi et al. [116] propose Fast Retina Keypoint (FREAK) (Figure 2.22 right), which is a descriptor that uses an overlapped fine-to-coarse pattern around the interest point.

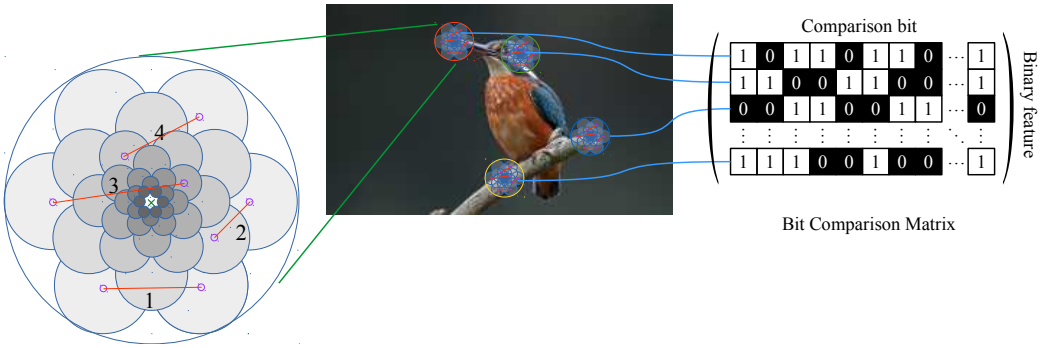


Figure 2.24: Bit Comparison Matrix. From a set of interest points, we calculate their binary features and store them in a matrix. This matrix is an array whose size is $N \times D$, where N is the number of features and D number of pair comparisons.

As we can see from Figure 2.22, we can generate thousands of bits by comparing the regions using an all-vs-all pairs scheme (see Figure 2.23 for a toy example); therefore we have to limit the comparisons by selecting a few pairs. Choosing those pairs is an initial stage because we need to find out which ones are the most descriptive. This procedure is known as best-pair seeking [115, 116]. This seeking implies to determine which pairs generate the most distinctive features. To this end, we require to calculate a bit comparison matrix from a set of interest points, as Figure 2.24 illustrates. This matrix comprises N number of features generated by D number of pair comparisons. Thus each row represents a feature and each column

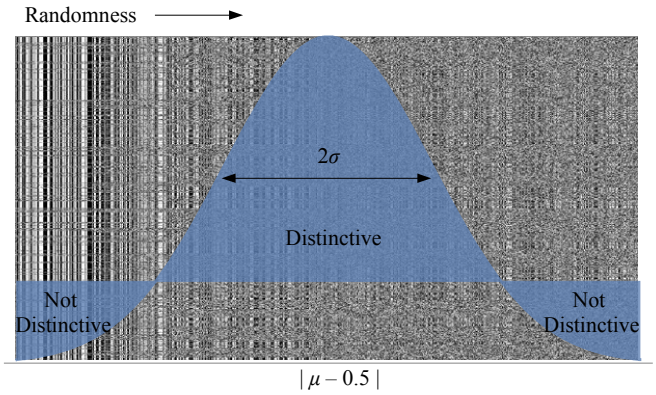


Figure 2.25: Best-pair seeking process. Each column represents a comparison bit generated by one pair. The most discriminant pairs appear to be within the range 2σ.

a comparison. The seeking procedure is as follows using the bit comparison matrix:

1. Compute the mean for each feature dimension (comparison bit), where we associate the columns with dimensions and rows with features.

2. Order the columns according to the mean. The first columns are the one with means closest to 0.5.

3. Calculate the correlation between all columns (all-vs-all matrix).

4. Select the best column (i.e. the one with the highest variance) and iteratively select the remaining columns that are less correlated to the best column using the all-vs-all correlation matrix.

After this procedure, the pairs are sorted from the most to the less descriptive (see Figure 2.25 produced by FREAK patterns). The rest is selecting a number of those pairs to create the binary feature.

Video Features

Ma and Cisar [118] propose binary features for event detection and is one of the earliest work in video analysis that successfully incorporates binary features to the spatio-temporal domain. They propose a variant of LBP for the spatio-temporal domain using dynamic textures which achieves competitive results in video surveillance. Specifically, they generate binary features by comparing the pixel intensities of the centre pixel of the support regions compared with its local neighbours.

Raginsky and Lazebnik [119] propose a hashing technique to map the double precision features to the binary domain in order to speed up computations in subsequent steps; the method is called Local Sensitive Hashing (LSH). This mapping naturally requires firstly to calculate the orientation-based descriptors. Therefore, its total complexity is equal to the double precision features complexity plus the complexity of the mapping process.

Inspired by LBP, Kliper-Gross et al. [120], Yeffet and Wolf [121] propose Motion Interchanges Patterns (MIPs). Matching pixel intensities in local neighbourhood has been proven to be an efficient technique to classify actions, despite it is not a feature itself defined by a STSR. Specifically, they propose to efficiently capture the motion in the local spatio-temporal region for each pixel in the video by using three fixed values $(-1, 0, 1)$ called trinary patterns. Thus, the MIPs encode no STSR but the whole video sequence into a binary feature. They associate the concatenation of different MIPs with actions, e.g., handclapping, running, walking. This system is an practical approach to recognise actions with outstanding computational efficiency.

Discussion and Drawbacks

Binary-based descriptors for images have been shown to have good performance in object detection. This fact leads to thinking immediately: why not just directly extend the 2D binary-based formulations to the spatio-temporal domain to create a new binary video descriptor? Let us discuss some essential aspects.

LBP: This technique assumes that capturing pixel intensities comparisons generates distinctive features. The original work is tested taking a pixel and see which neighbours have lower or higher pixel intensities. This procedure, unfortunately, is notably sensitive to noise when the compared pixels are distant. Hence, comparing not isolated pixels but summed patch intensities lead to more distinctive features because of their robustness to noise. Therefore, LBP needs necessary adjustments.

Existing Patterns: One can verify that extrapolating the 2D patterns (e.g. FREAK, BRISK) to the spatio-temporal domain lead to very poor results regarding accuracy in action recognition. We observe the following aspects that could explain this problem.

- As the inner regions (see Figure 2.22) are progressively smaller, the comparisons between the small and the large ones generate practically random bits making the descriptors very sensitive to noise. Perhaps this explains their low

performance.

- By extending the 2D pairs to 3D pairs, the number of compared pairs significantly increases (in the order of thousands), making the best-pair selection very computationally expensive because of the all-to-all correlation matrix used to find the best pairs.

- The best-pair selection is self-defeating because it in line with at least one of the Golomb's postulates of random series.

  In the best-pair seeking, we sort feature columns according to their correlation and high variance (or mean greater than 0.5), but the first Golomb postulate states that a random series contains approximately the same number of 1's and 0's. For example, assume that a particular column contains the series $1010101\ldots$ whose mean is 0.5. According to the seeking procedure, we have to select this column even though the feature in that column is obviously not descriptive.

  During the best-pair seeking, one can observe that after the procedure's correlation iteration, the process outputs pairs that produce very random bits. We can obtain better results by only selecting the bits inside the range 2σ having the mean of the columns as the input variable of a normal distribution. Therefore, random bits generate low variant features which are not highly descriptive. This aspect contradicts the best-seeking pair criterion.

- Selecting the most descriptive bits is a complex task if we consider the Golomb postulates. Ideally, the generated bits should not satisfy any of the postulates; one can prove that the most random bits are the ones with less discriminating capacity and also the ones that closely match the postulates. Thus the best-pair seeking procedure should not be consistent with the postulates and contrary it is.

- One of the aspects we found inconsistent with the selection criteria is that according to the authors [114, 116], more bits should enhance performance. However, one can verify that more bits tend to create confusion instead. After 128 bits of the 2σ range, one can verify the accuracy starts to decrease. This aspect lead us to conclude that more bits do not necessarily enhance accuracy.

These exposed aspects require attention in order to generate robust binary-based features for video analysis.

2.3.3 Binary vs Double Precision Features

The main drawback of double precision features is that they are high-dimensional vectors [6, 7, 11]. High-dimensional vectors usually result in high storage demands. As a consequence, further processing in action recognition using these descriptors, e.g. clustering and matching, involve long computational times [113, 116]. The data format they need also results in excessive memory demands, for instance, [7, 11] descriptors produce features that require more storage than that needed to store the video itself, sometimes, more than a thousand times. We can list the four main drawbacks regarding they computational efficiency.

1. Every pixel within the STSR requires computing its orientation.
2. The resulting orientations require binning and frequency counting.
3. Storing those features requires double data precision.
4. The dimensionality of the resultant feature vector is considerably high.

These aspects inevitably hinder computational speed and storage efficiency. Hence, it is essential to develop low-dimensional video descriptors that do not considerably increase computations and storage demands.

Binary features for action recognition are an effort in the pursuit of lower computational complexity. This low computational complexity is the primary motivation for various image binary-based descriptors [113–116, 122]. Binary-based features rely on one core operation, which is the comparison. This operation is calculated remarkably fast, and its result requires only fractions of memory for being stored. Therefore, binary features are an attractive alternative for video processing to speed up feature extraction or reduce computational demands.

This page intentionally left blank.

Chapter 3

Graph-based Abnormal Event Detection

This chapter starts with some essential aspects of the state-of-the-art. Out of the abnormal event detection techniques, Trajectory-based (Section 2.2.1) and No Trajectory-based (Section 2.2.1), I explore the latter. The reason to explore this technique is that tracking objects in the scenes is itself a challenging problem [83, 123, 124]. Not only do background conditions [123] complicate object's segmentation but also the overlapping with other moving objects [83, 124]. As a point of fact, in video surveillance crowded scenes [72, 77] are very common making tremendously challenging tracking objects. This complication is the reason behind proposing no-tracking approaches [3], and why no-tracking-based methods achieve better performance [4, 97]. Nevertheless, we should highlight that the object's displacement is undoubtedly a rich source of motion information. However, solving the problems related to the very core of tracking are out of the scope of the interest of this thesis. Instead of dealing with the tracking problem I employ essential concepts regarding no-tracking abnormal event detection to elaborate new formulations based on graph theory.

We propose a method to detect abnormal events in the video without minding the processing times. To this end, graph-based methods [4, 45, 76] have been shown promising results; thus we decide to analyse a video sequence as a fully connected graph.

Roshtkhari and Levine [4] propose to use as motion source the temporal gradient. This motion source proved to be efficient in terms of speed and accuracy, however, taking only pixel intensity is not very descriptive as [2] reveals. Thus we decide to **enhancing the local descriptor** using this motion source.

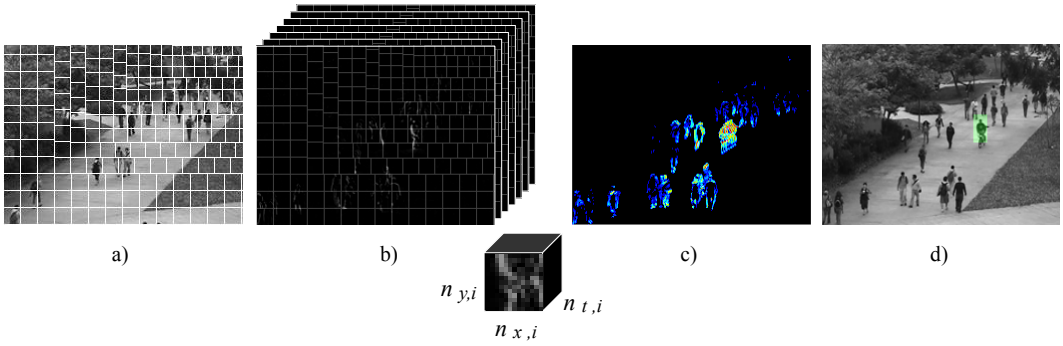


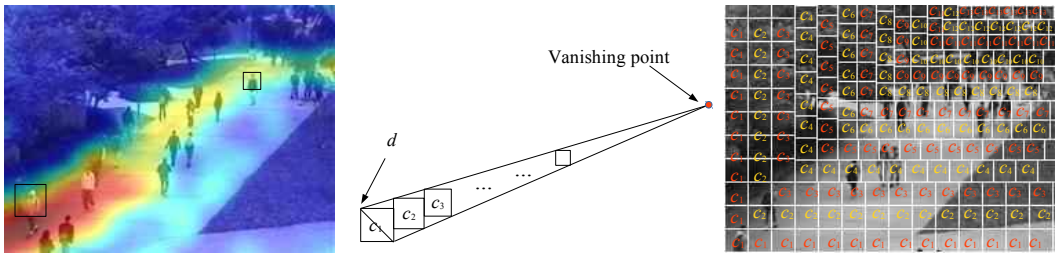
Figure 3.1: The proposed method. **a)** The perspective grid to extract video volumes. **b)** The temporal gradient detects motion, and the perspective grid defines the video volumes. **c)** We merge each video volume using the maximum operator to compute the WMD. A set of connected graphs are generated to capture moving objects. **d)** We use the graphs' properties to detect abnormal movement.

Thida et al. [45], Kim and Grauman [76] propose to analyse the video as a fully connected graph to detect abnormal events by examining the local features' composition. Thus, we represent STSRs as fully connected graphs, assuming that the local descriptor enhances graph-based methods.

Contributions: We propose a graph-based method to detect video anomalies by analysing video volumes. Our approach employs the Wake Motion Descriptor (WMD) to describe the trace left behind by a moving object in the scene. The method takes into account the perspective of the scene to construct the video volumes. Specifically, it uses varying-size volumes according to the relative position of the regions they describe in relation to the camera's location. By using a probabilistic inference process, the method analyses and classifies video volumes. Those video volumes with not previously seen wakes are deemed to be abnormal. The main contributions in this chapter lie therein the gathering of motion information. We can summarise them as:

- We improve feature extraction via WMD.
- The STSR are of variable size compensating the perspective of the scene.

This chapter has four sections. Section 3.1 presents details of the proposed graph-based method. Section 3.2 shows results of the proposed method using the UCSD dataset. Section 3.3 gives discussions and analysis of the results obtained. We conclude this chapter in Section 3.4.



a) Average speed **b)** One – point Perspective **c)** Perspective Grid

Figure 3.2: **a)** Optical flow of a scene as it increases from blue to red for those regions close to the camera. **b)** Starting from an initial cell c_1 of size $n_{x,1} \times n_{y,1}$, we generate smaller cells as they approach the vanishing point of the scene. **c)** Using set D, which contains the diagonal size of all cells, $D = \{\lfloor d_1 \rfloor, \lfloor d_2 \rfloor, \ldots, \lfloor d_n \rfloor\}$, the perspective grid is generated by sweeping the scene in the vertical and horizontal direction starting with cell size c_1, which is located at the point farthest from the scene's vanishing point.

3.1 Proposed Graph-based Inference Framework

Figure 3.1 graphically summarises the method. First, the gradient along the whole video sequence in the time direction detects motion. Then we extract video volumes considering the perspective of the scene so that video volumes depicting regions close to the camera have bigger size than those far from it. Later we merge the frames comprising each video volume using the maximum operator to compute the WMD. This descriptor is then used to generate a set of connected graphs. Finally, the graphs' properties are used to detect abnormal movement. The next sections describe each step in detail.

3.1.1 Perspective Grid and Video Volumes

When capturing a scene, we commonly find that the camera's field of view is at an angled position from the principal plane of movement. In this case, objects appear to move faster as they approach the camera, even if they move at a constant speed. Figure 3.2a illustrates this effect, where the optical flow's magnitude increases for those regions closer to the camera. Moreover, objects close to the camera appear to be bigger than those located far from it even if they are the same size. Within the context of anomaly detection using video volumes, the relative change in an object's speed and size introduced by the camera's angle of view may have a detrimental effect on the extracted features.

The apparent object size changing problem may be solved by scaling objects, in size and speed, as they move closer to the camera in the scene; however, any scal-

ing method, e.g. nearest-neighbour interpolation, requires object tracking. Thus, we explore a solution to this problem.

In this thesis, unlike common grid-based methods, e.g. [81], [50], we propose to create video volumes of different sizes in the XY plane. These sizes are according to their position relative to the camera's location. In other words, the size of the XY plane of volumes increases as they capture regions that are closer to the camera's location. To create volumes of different size, we build a *perspective grid* for the whole sequence, which comprises cells that define the size of the video volumes in the XY plane. Figure 3.2c illustrates this construction, where those cells depicting regions near the camera are bigger than those located far from it.

To construct the perspective grid we detect the vanishing point p of the scene. In this case, we detect it manually. The process is as follows: starting from an initial cell c_1 of size $n_{x,1} \times n_{y,1}$, we make cells smaller as they approach the vanishing point, Figure 3.2b illustrates this process. For each generated cell, we store its diagonal size, $\lfloor d_k \rfloor$, in the set D:

$$D = \bigcup_k \lfloor d_k \rfloor. \tag{3.1}$$

Then, we use the elements in D and the location of p to sweep the scene vertically and horizontally to define the volume sizes in the XY plane (see Figure 3.2c). We obtain a video volume v_i by selecting the subregion denoted by cell c_i of size $n_{x,i}$ $\times$ $n_{y,i}$ over $n_{t,i}$ frames along the sequence, as Figure 3.1b illustrates. In this thesis, $n_{t,i}$ is constant for all video volumes.

3.1.2 Wake Motion Descriptor

The proposed method captures the trace left behind by a moving object in the scene, i.e. its wake. This trace provides information about the object's motion characteristics. We expect to have different wakes from different objects moving at different speeds, e.g. a boat crossing a lake fast or slow. Therefore, we can classify an object based on its wake.

Figure 3.3 shows the wake generation from a video sequence. To calculate the scene's wake, we first compute the gradient in the time direction along the sequence:

$$M_j = I_{j+1} - I_j, \tag{3.2}$$

where j denotes the index of frame I. For a sequence of J frames, sequence M comprises $j-1$ gradient frames. Next, we merge n_t mini-frames inside each video

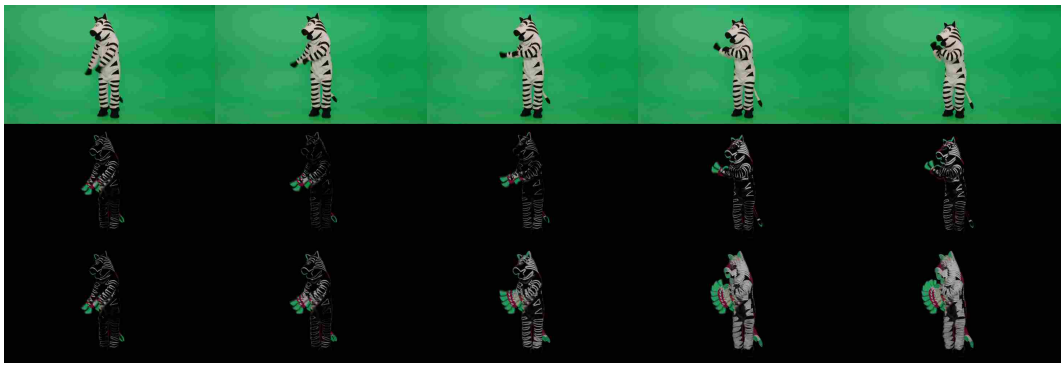


Figure 3.3: Motion source to extract wakes. (Top) A video sequence and its (middle) temporal gradient. (bottom) The wake calculated from the temporal gradient.

volume v_i into a 2D array of size $n_x \times n_y$ to generate the corresponding wake w_i:

$$w_i = max_{\Delta t}[v_i], \tag{3.3}$$

where $max_{\Delta t}$ denotes the maximum operator in the time direction. Then, we analyse the texture in the wake w_i using the SFTA descriptor.

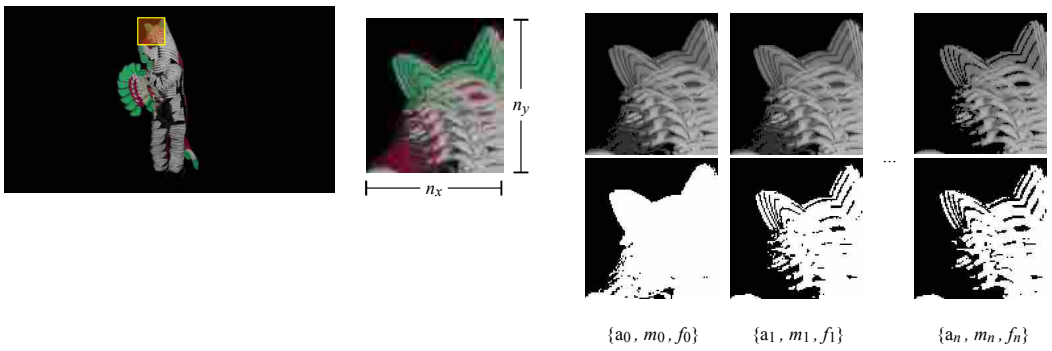


Figure 3.4: Example support region used by WMD to extract the SFTA feature. This descriptor binarises the Image using different thresholds. For each binarisation, it calculates the area, mean and fractal dimension.

The SFTA descriptor requires decomposing an image into a set of binary images (see Figure 3.4). From those binary images, the fractal dimensions of the resulting regions are computed to describe segmented texture patterns. SFTA generates a feature d, comprising the area, mean and fractal dimension of the binarised

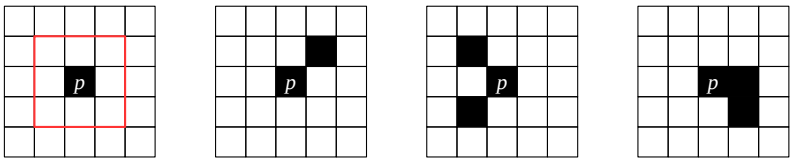


Figure 3.5: a) N_8 pixel connectivity of pixel p with its immediate neighbours (enclosed by the red box) . (b - d) one or more pixels make p 'connected'.

image [125]. The binarised image B, comprising N pixels, is calculated as:

$$B(x,y) = \begin{cases} 1, & \epsilon_0 \leqslant I(x,y) \leqslant \epsilon_1 \\ 0, & \text{otherwise} \end{cases}, \tag{3.4}$$

where ϵ_0 and ϵ_1 are empirical thresholds. We calculate the area a and mean m as follows:

$$a = \frac{1}{N}\sum_{x,y} B(x,y) \;\; m = \frac{1}{N}\sum_{x,y} I(x,y). \tag{3.5}$$

The fractal E of the image is as follows:

$$E(x,y) = \begin{cases} 1, & \exists (x',y') \in N_8\left[(x,y)\right] \\ 1, & B(x',y') = 0 \wedge B(x,y) = 1 \\ 0, & \text{otherwise} \end{cases}, \tag{3.6}$$

where N_8 is the eight pixel connectivity (see Figure 3.5). The fractal dimension f is calculated as:

$$f = \frac{\sum_s \log C^s_{x,y}}{\log s}, \tag{3.7}$$

$$C^s_{x,y} = \sum_{\substack{x_0 \leq x \leq x_1 \\ y_0 \leq y \leq y_1}} E(x,y), \tag{3.8}$$

where $s = (x_1 - x_0)(y_1 - y_0)$ is the scale. Using a set of normal scenes, i.e., training data, we construct a GMM with K components using the features $d_i \in \mathbb{R}^D$ and determine the corresponding parameters θ using the EM algorithm. Thereby, for each feature we have:

$$\mathcal{N}(d_j|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} \exp\left(-\frac{1}{2}\left(d_j - \mu\right)^\top \Sigma^{-1}\left(d_j + \mu\right)\right), \tag{3.9}$$

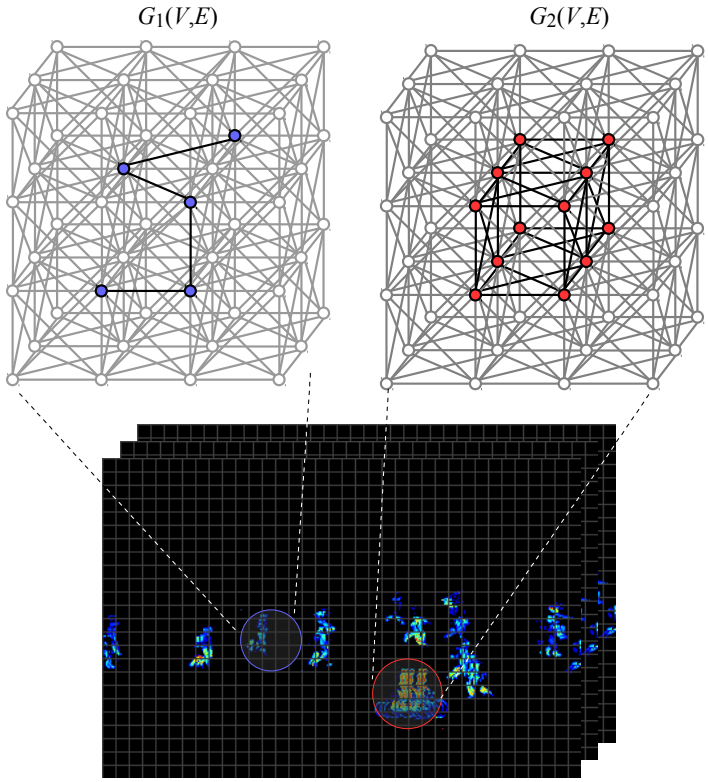


Figure 3.6: Two graphs are representing the characteristics of two objects' motion in the scene. The graph in blue corresponds to a pedestrian's motion while the one in red corresponds to a bicycle's motion.

where $\mathcal{N}(\mu, \Sigma)$ denotes a Gaussian distribution with mean μ and covariance matrix Σ. Equation (3.9) is used to determine whether incoming features have a high probability to appear accordingly to the previously seen ones.

3.1.3 Graph Formation

We can represent small space-time regions interactions using a set of connected graphs [45]. A graph $G_i(V, E)$ is created comprising all those connected video volumes in space-time and whose feature d meets the following criterion:

$$\epsilon \leq -log[\prod_{k} p(\theta_k|d_j)], \tag{3.10}$$

where ϵ is a threshold and $p(\theta_k|d_j)$ is the model of Equation (3.9). In other words, we connect video volumes v_i and v_j in space-time if their respective features satisfy

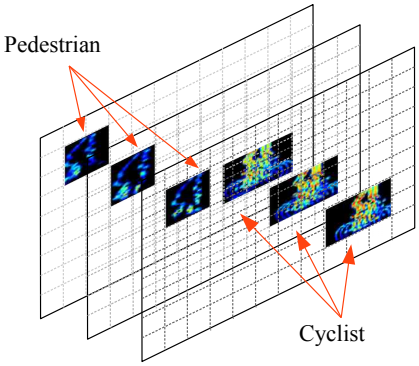


Figure 3.7: Wakes: the trace left behind by two different objects from the UCSD dataset. (left) Pedestrian (right) Cyclist.

Equation 3.10, then an undirected edge is generated between v_i and v_j.

From the graphs (see Figure 3.6), we infer the object's size, speed and motion duration in the scene. To this end, we build a model with the most common graph's properties from a set of normal frames. Figure 3.7 shows the wake left behind by two objects coming from abnormal frames. The wakes are consistently different for objects of different nature.

3.1.4 Anomaly Detection

We can classify moving objects based on their corresponding graph's properties, e.g., number of vertices $|V|$ or edges $|E|$, diameter, connectivity, density, etc. In this thesis, we detect abnormal motion patterns by considering each graph's edge size $x_i^{|E|} = G_{i,|E|}$ and diameter $x_i^D = G_{i,D}$. The former corresponds to the longest path s in the graph for any two vertices u and v:

$$G_{i,D} = max_{u,v} s(u,v). \tag{3.11}$$

We classify different graphs using a normal distribution $N(\mu, \sigma)$. In the training stage, we estimate the mean, μ, and variance, σ of a set of U normal graphs. The μ and σ of the edges' size are as follows:

$$\mu_{|E|} = \frac{1}{U}\sum_{i=1}^{I} x_i^{|E|} \quad \sigma_{|E|} = \frac{1}{U}\sum_{i=1}^{I}(x_i^{|E|} - \mu_{|E|})^2. \tag{3.12}$$

For the diameter of the normal graphs, μ and σ are:

$$\mu_D = \frac{1}{U}\sum_{i=1}^{I} x_i^D \quad \sigma_D = \frac{1}{U}\sum_{i=1}^{I}(x_i^D - \mu_D)^2. \tag{3.13}$$

The probability that a graph G_i fits the model is given by:

$$p(G_i) = \prod_{k=\{|E|,D\}} \frac{1}{\sigma_k\sqrt{2\pi}} e^{-\frac{(x_i^k-\mu_k)^2}{2\sigma_k^2}}. \tag{3.14}$$

The graph G_i is deemed to be normal if the following assumption is satisfied:

$$\gamma \geq -\log[p(G_i)]. \tag{3.15}$$

In this way, we can detect abnormal graphs associated with abnormal STSRs that depict unusual moving objects.

3.2 Experimental Results on UCSD

We use the popular UCSD pedestrian datasets [126] to test the proposed method. The dataset contains footage from two pedestrian side-walks where abnormal events occur. The first scene contains 34 normal video clips and 36 abnormal video clips; the second scene contains 16 normal video clips and 14 abnormal video clips. Both scenes contain different crowd densities and non-pedestrian moving objects, namely bikes, small cars, skaters and wheelchairs, which are deemed to be abnormal.

We use the normal videos of this dataset as training data to derive the probabilistic models in Equation 3.9 and 3.14. *Empirically* Equation 3.10 and Equation 3.15 have values of $\epsilon = 7.5$ and $\gamma = 2.3$. For the model in Equation 3.9, $K = 3$ Gaussians models. We selected these values because they provide the best detection accuracy during the evaluations.

We compare the proposed graph-based method against many state-of-the-art approaches. There are two ways to this end, either at a pixel or frame level detection. This second implies considering the frame abnormal regardless of the abnormal pixel locations. For the former case, we have to compare the abnormal frames against the ground truth at a pixel level to locate the position of the unusual event.

We use the Equal Error Rate (EER) to rank our method. This metric describes the rate of misclassified frames; it is smaller when the system correctly identifies abnormal frames. If the EER increases, the cause could be (I) the system

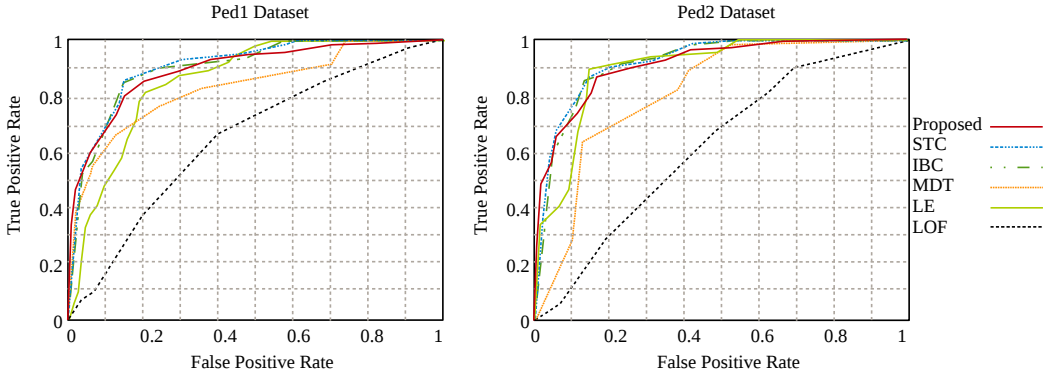


Figure 3.8: Frame level ROC comparison of the proposed graph-based method and state-of-the-art methods for the UCSD pedestrian datasets. [4] STC, [10] IBC, [45] LE, [77] MDT, [1] LOF

is not detecting them or (II) those identified frames are not abnormal.

For the first experiment (frame level), we consider a frame to be abnormal if it contains at least one anomalous pixel. For this case, Figure 3.8 shows the detection accuracy of the state-of-the-art and the proposed method via the Receiver Operating Characteristics (ROC) curve. Note that the proposed approach achieves very similar performance to that attained by [4, 10], but with shorter computational time and less number of training frames.

For the second experiment (pixel level), we consider a frame to be abnormal if it contains at least 40% of the abnormal pixels given by the ground truth. Otherwise, as [77] suggest, we consider the detection as a false alarm.

Table 3.1: EER for the UCSD pedestrian datasets for various methods and corresponding no. of required training frames

Method	Dataset	Frame Level (%)	Pixel Level (%)	No. of Training Frames
Proposed	Ped1	19	26	200
	Ped2	16	25	180
Roshtkhari and Levine [4]	Ped1	15	27	200
Spatio-Temporal Compositions (STC)	Ped2	13	26	180
Boiman and Irani [10]	Ped1	14	26	6800
Inference By Composition (IBC)	Ped2	13	26	2880
Thida et al. [45]	Ped1	13	22	1000
Laplacian Eigenmap (LE)	Ped2	22	31	1000
Mahadevan et al. [77]	Ped1	25	58	6800
Mixture of Dynamic Textures (MDT)	Ped2	24	54	2880
Adam et al. [1]	Ped1	38	76	6800
Local Optical Flow (LOF)	Ped2	42	-	2880

Table 3.1 tabulates the EER at pixel and frame levels, and Figure 3.9 shows visual results. Results in Table 3.1 show that the proposed method outperforms

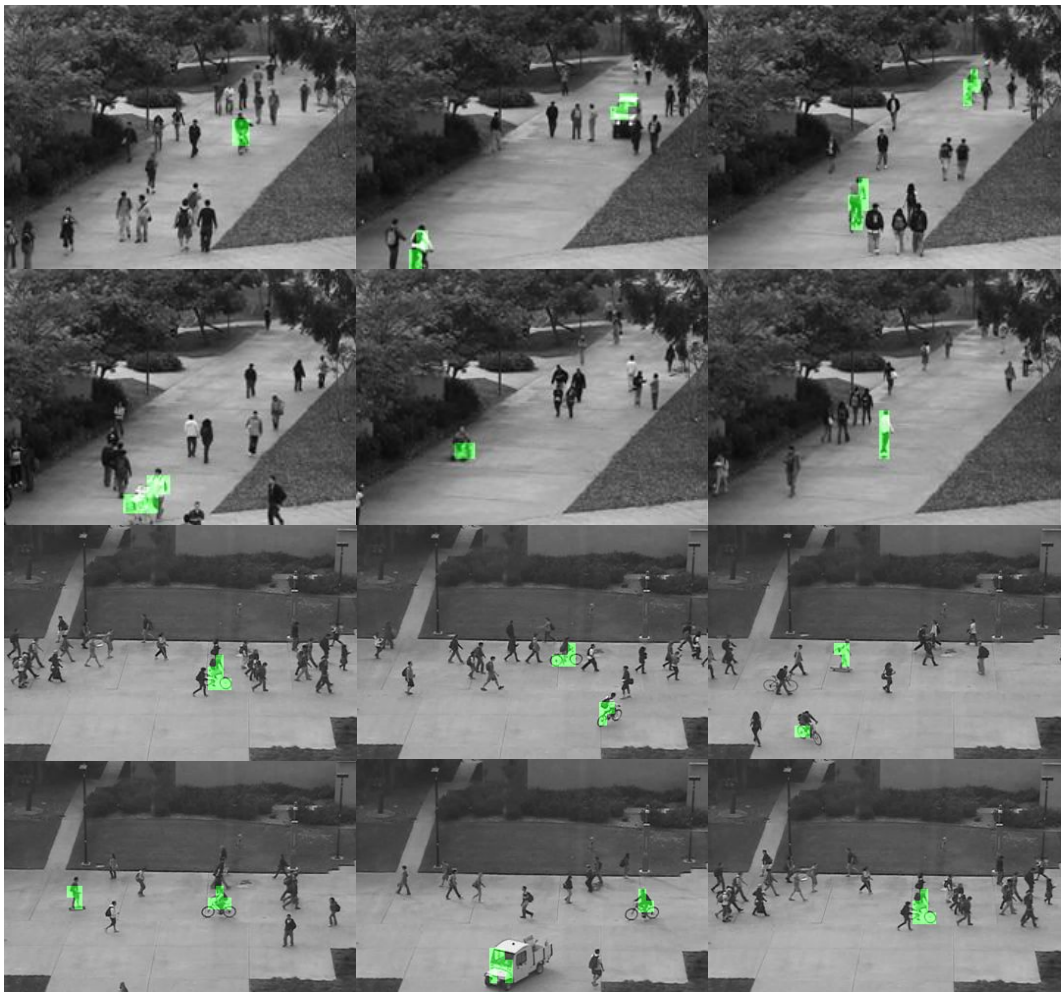


Figure 3.9: Some samples of abnormal frame detection for the UCSD pedestrian datasets. Row (1-2) depicts scenes from the Ped1 set and (3-4) from the Ped2 set. Highlighted in green the abnormal regions. The proposed method detects cyclists, skaters, wheelchairs, and small cars.

some of the state-of-the-art approaches on the UCSD pedestrian datasets regarding detection accuracy at a pixel level.

Note that the proposed method requires the same number of frames in the training stage as [4], which is considered the state-of-the-art due to the small numbers of frames necessary to train the method.

The proposed method also attains competitive computational times as reported in Table 3.2. We run all tests using a PC: Intel Core Quad with 2 CPUs and 7.8GB of RAM without parallelising the code.

We notice that although the proposed method is able to detect all non-

pedestrian movements in both datasets, it is sensitive to synchronised crowd movement in the same direction; e.g. a group of pedestrians walking in a synchronised fashion in the same direction. This effect is mainly due to occlusions among objects. In these cases, the WMD confuses the crowd as a combination of objects moving in the same direction at the same speed with a big moving object.

Table 3.2: Computational time per frame

Method	Dataset	Time (seconds)
Thida et al. [45]	Ped1	0.0065
	Ped2	0.0065
Roshtkhari and Levine [4]	Ped1	0.19
	Ped2	0.22
Proposed	Ped1	1.22
	Ped2	1.55
Mahadevan et al. [77]	Ped1	21
	Ped2	29
Boiman and Irani [10]	Ped1	69
	Ped2	83

We also evaluate the benefits of using the proposed perspective grid. Table 3.3 reports the detection accuracy at a frame and pixel level of the proposed method when the perspective grid is used (PG) and not used (No PG). Let us recall that we deem a frame to be correctly identified as abnormal if the method detects at least 40% of the ground truth pixels.

Table 3.3: EER of the proposed method for the UCSD pedestrian datasets when using Perspective Grid (PG)

Method	Dataset	Frame Level (%)	Pixel Level (%)
Proposed-(PG)	Ped1	19	26
	Ped2	16	25
Proposed-(No PG)	Ped1	26	35
	Ped2	21	27

In both datasets, the proposed method can detect abnormal motion patterns in the whole frame regardless whether the moving objects are close or far from the camera's position. The framework identifies the unusual objects as soon as they appear in the scene.

Note that the benefits of using the perspective grid are more significant for dataset Peds1 than those for UCSD dataset Peds2. This enhancement is due to the camera position in dataset Peds1 from the principal plane of movement, as Figure 3.9 illustrates.

3.3 Discussions

The proposed method achieves competitive performance detecting anomalous events; the next step is to identify unusual activity in real video surveillance. However, at this point, we have to recall an important aspect. To create a practical application of the proposed graph-based proposed method, we have to reduce the computational times of the core steps, i.e. SFTA and graph properties. To solve the long computational times, as Table 3.2 shows, the method requires processing frames at least **50**× faster to be online performing. Specifically, the graph properties extraction is very computational expensive considering the spatio-temporal neighbourhood exploration of each vertex for all paths. We observed that each vertex requires an all-vs-all subpath search leading to an $O(V^3)$ complexity algorithm. Further optimisation of this algorithm may lead to an $O(E^2 + Vlog(V))$ algorithm [127]. However, it is still too high to reduce computations as needed.

Another complication is that the SFTA descriptor requires computational times of c.a. **15ms** per STSR – thus the complexity is also very high – forcing us to explore a faster feature extraction method. Specifically, this descriptor requires each pixel N_8 connectivity, analysing the fractal dimension for each binary image $B(x, y)$ over different scales s giving an $O(n^4)$ algorithm.

Unfortunately, the previously mentioned aspects hinder our method's speed performance. All these aspects lead us to consider new formulations to represent the video structure. Mainly, we have to capture spatio-temporal compositions not requiring such a high complex graph search, and we have to encode STSRs faster. These aspects are the motivation of the next chapter.

3.4 Conclusion

This chapter proposed a graph-based method for anomaly detection in video. This approach is based on statistical inferences using video volumes and a wake motion descriptor. In order to compensate for the camera's position and the underlying perspective of the scene, the method uses a perspective grid to define the size of video volumes according to their proximity to the camera.

Based on the wake motion descriptor, the method constructs graphs of spatiotemporal connected video volumes that describe moving objects. Those moving objects with graphs whose characteristics do not fit a probabilistic model are deemed to be abnormal. We tested the method against several state-of-the-art methods. Results show that it provides a competitive detection accuracy.

This page intentionally left blank.

Chapter 4

Online Abnormal Event Detection

As we stated in the previous Chapter 2 abnormal events can be detected under two different scopes: accuracy first (Section 2.2) and speed first (Section 2.3). The former, despite its better precision, is impractical for real scenarios. This Chapter proposes a method to detect abnormal events in real-time while maintaining high detection accuracy. The graph-based method proposed in 3 motivates this Chapter. Even though the graph-based method attains good accuracy, it is unsuitable for practical scenarios as Section 3.3 reveals.

As we can see from Section 2.2.3, there is a clear trade-off between detection accuracy and processing times in video anomaly detection. The challenge is then to appropriately balance this trade-off by employing few, but highly descriptive, features to attain online performance with a competitive accuracy.

Therefore, in this chapter, I propose an online event detection method comprising low complexity techniques. The proposed approach does not require an optimised version to reduce the computational times as directly addresses the time-constraint problem. Our **contributions** as are follows:

1. We incorporate optical flow, temporal gradient and foreground occupancy as motion sources in a **dual-inference** mechanism for abnormal event detection.

2. The proposed framework uses a **multi-scale scanning** technique compensating the apparent change of object's size and speed. Thus, we detect events close and far from the camera.

3. We **combine** the previous two points along with efficient HMM and STIP techniques into an online method.

The core of our framework is to generate features from a limited number of STSR efficiently. Henceforward, we refer to the proposed online method as Abnormal Detection with Compact Sets of Features (ADCSF). The ADCSF balances the speed-accuracy trade-off by using a compact set of optical flow and foreground occupancy highly descriptive features. A novel cell structure achieves this balance. Features are extracted only from those cells that are deemed to be relevant to the analysis. The main differences between existing approaches and the ADCSF are as follows:

- Although other cell structures exist, e.g. [50, 51, 81], the ADCSF comprises a novel fine-to-coarse cell structure that is computationally efficient and processes cells of multiple sizes. The latter helps to take into account the intrinsic camera-object distance in the analysis.

- Instead of employing the commonly used convolution-based STIPs, e.g. [98, 99], the ADCSF comprises their binary counterpart FAST [21], see Section 2.1.3. FAST STIPs therefore also contribute to attaining online performance.

- The ADCSF is fed with two motion sources to extract high descriptive features achieving online performance for the first time. It processes optical flow and foreground, both of which have been successfully used separately in the past [4, 81].

- The ADCSF successfully processes optical flow features, which are known to be highly computationally complex [4, 97] for online performance. We propose tackling this problem by extracting features only from those cells that are relevant to the analysis.

The core of the ADCSF is to efficiently describe the events in the scene without computing a significant number of features. Hence, the ADCSF focuses on an efficient scene representation using a compact set of features.

The rest of this chapter is organised as follows: In section 4.1, we describe the ADCSF. In section 4.2, we present experiments and discussions. We conclude this chapter in section 4.3.

4.1 Proposed Online Inference Framework

Figure 4.1 graphically summarises the ADCSF. In the Training Stage, first, we construct a cell structure for the whole scene to define the STSRs (Section 4.1.1)

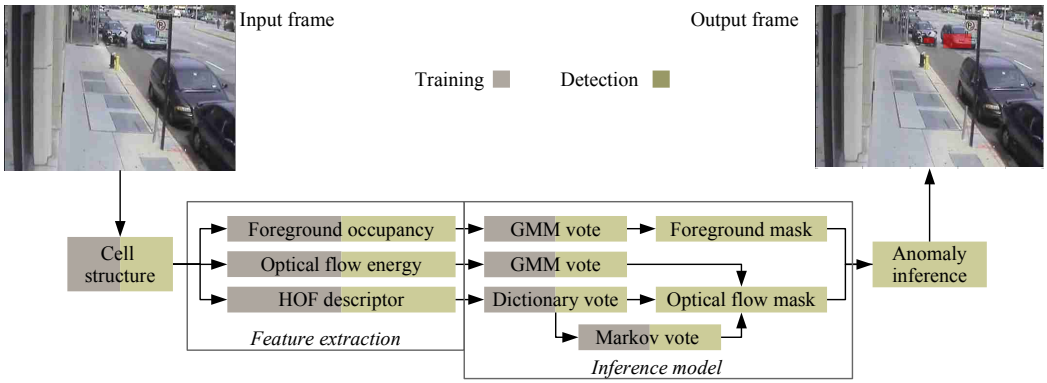


Figure 4.1: The ADCSF. We build a cell structure to define the STSR. We extract features from foreground occupancy and optical flow to construct four models. In the detection stage, we use these models' outputs to create two inference likelihood masks to detect of anomalous events.

to be analysed (Section 4.1.2). Then, the extraction from a compact set of features takes place, i.e., from a limited number of STSRs. We extract these features from foreground occupancy and optical flow (Section 4.1.3). The compact set of features is analysed to construct various models. In the Detection Stage, we use the models to detect anomalous video volumes (Section 4.1.4). Finally, an inference mechanism that considers the local spatio-temporal neighbourhood of cells is used to identify abnormal events (Section 4.1.5.). This section is organised as follows:

- Section 4.1.1 explains the motion sources of the ADCSF.
- Section 4.1.2 details the cell structure necessary to encode the STSR.
- Section 4.1.3 explains feature extraction process.
- Section 4.1.4 describes the probabilistic model computed from the extracted features.
- Section 4.1.5 explains the inference method used to detect abnormal events.

4.1.1 Motion Sources

The initial step of the abnormal inference frame is to define the motion sources. There are several motion sources proposed in the literature, for example foreground occupancy, e.g. [81], temporal gradients, e.g. [4, 47] and optical flow, e.g. [87, 88]. Figure 4.2 illustrates these three motion sources.

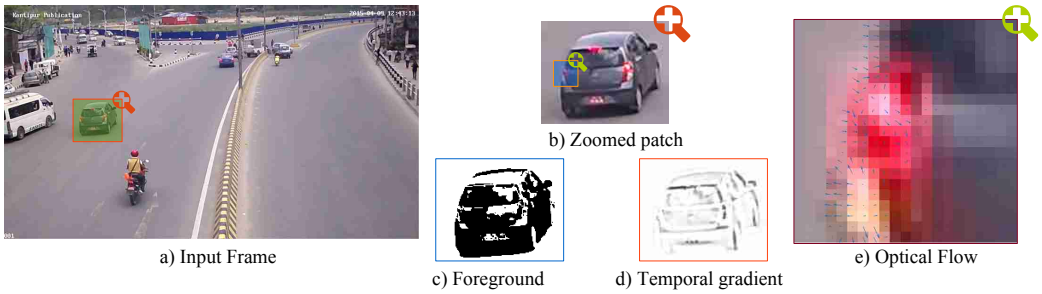


Figure 4.2: **a)** Input frame. Three motion sources are displayed: **c)** foreground occupancy, **d)** temporal gradient and **e)** optical flow. There later is zoomed in from **b)** for visualisation purposes.

Modelling the foreground occupancy has significant advantages (see Figures 4.2c and 4.2e). One is that long-term events, e.g. loitering, can be easily detected with this motion source [1]. For instance, we can define loitering not for what the subject does but for what the subject stops doing (inactivity). Since the pixels that capture the subject do not change, pixel-change based motion such as the temporal gradient or optical flow, does not generate any useful information. In order words, when objects do not move they merely appear to be part of the background. Another advantage is that foreground occupancy can also be used to define the object's size [81]. Specifically, we can estimate the object's size in a scene without requiring segmentation.

Temporal gradient (see Figure 4.2d) is also useful as a motion source. One of its most important advantages is the time required for its calculation [4]. Regarding descriptiveness, it achieves competitive accuracy in action recognition [2]. It also achieves competitive performance in action recognition while attaining short processing times [4, 47, 97].

The majority of abnormal event detection methods employ optical flow. One of the reason is because it is very accurate for action recognition [128]. Using the STSRs to extract optical flow features make this motion source highly reliable, one example is the HOF descriptor [106] as described in Chapter 2. The more descriptive the features are, the more accurate the inference method is [98]

As discussed, all these motion sources are useful to detect abnormal events. Combining them is a significant contribution of this chapter because these sources have shown advantages and we propose to make use of all of them in a single framework. Specifically, we propose to use optical flow and foreground for feature extraction and temporal gradient to detect STIPs.

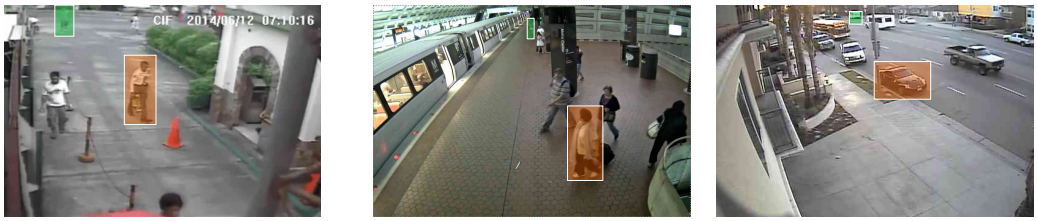


Figure 4.3: Sample frames illustrating the objects' apparent size change due to the camera's position. The object boxed in orange seems to be larger and move faster than those objects boxed in green.

4.1.2 Cell Structure

We have to define the STSRs to extract features from motion sources. Considering the insights of Section 2.2, we inevitably need to propose a computationally efficient multi-scale structure to define these STSRs. As Figure 4.3 illustrates, the STSRs must be larger as the objects are closer to the camera. In other words, we must compensate for the apparent change the object size due to the camera's position. Those STSRs relatively close to the camera provide more descriptive information than those located far from it. Therefore, taking into account the camera's position in the scene to extract features can significantly enhance performance [12, 81, 129].

To address the camera's position problem, we use a cell structure where cells are defined using a non-overlapping grid over the spatial domain. An equal-sized cell structure is intuitively not appropriate to deal with the camera's position and the associated scene's perspective, as it results in extracting features equally from all regions of the scene regardless of their location relative to the camera. A simple solution to compensate for the scene's perspective is to track objects as they enter and exit in it. However, object tracking is particularly challenging and computationally complex in crowded scenarios [83, 123, 124].

In this chapter, the assumption is that the camera is acquiring an unobstructed view of a scene and lies in a high position looking downwards. This assumption is valid for the majority of video surveillance cameras. When the camera captures downwards from an elevated position to acquire an unobstructed scene, the lower region of the scene then tends to be the one closest to the camera. Thus, we propose to create a grid with variable-sized cells where the largest cells capture the lower region of the scene (i.e., regions closest to the camera), and the the smallest cells capture the upper region of the scene (i.e., regions farthest from the camera). Large cells then provide more information to compute features.

The size of the cell is defined according to the frame size, see Figure 4.4.

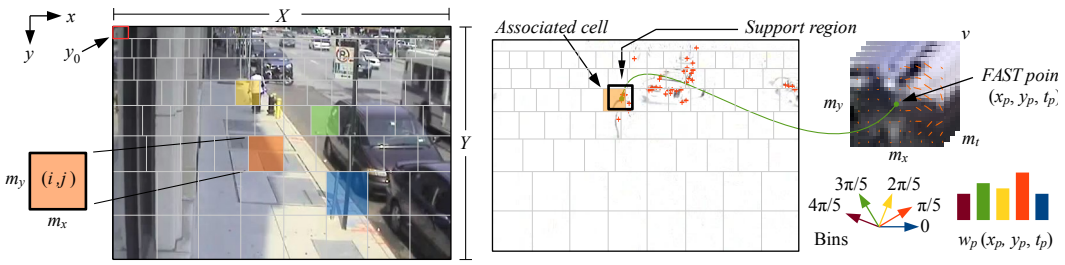


Figure 4.4: a) Example cell structure for a scene. Cells of different size are highlighted in different colors for illustration purposes. Largest cells correspond to regions closest to the camera. The cell c at position (i, j) has spatial dimensions of $m_x \times m_y$. b) we extract HOF descriptors from video volume v, which is the support region for the STIP at position (x_p, y_p, t_p). The size of v is determined according to the cell in which the spatial location (x_p, y_p) falls into.

Starting from the frame's top border, given the smallest support region where $y_0 = n_y$, the rest of the support regions vertically grow with a factor α. Let y_k be the vertical dimension of the kth cell as associated with its vertically adjacent cell, i.e., the $(k+1)$th cell, by:

$$y_k = \alpha y_{k+1}, \tag{4.1}$$

where $\alpha > 1$ is a *growth rate* that makes the $(k+1)$th cell larger than the kth cell. Thus, the vertical size of the frame, denoted by Y, can be expressed in terms of the recursive vertical dimension of each cell as follows:

$$Y = \sum_{k=0}^{n} \alpha^k y_0, \tag{4.2}$$

where n is the number of cells along the vertical dimension and y_0 is the vertical dimension of the smallest cell. To find n, initially we set y_0 to an initial value. Equation 4.2 can be easily transformed into its geometric series form:

$$Y/y_0 = \frac{\alpha^{n+1} - 1}{\alpha - 1}. \tag{4.3}$$

The value of n can then be calculated as follows:

$$n = \lfloor \log_\alpha (Y/y_0(\alpha - 1) + 1) - 1 \rceil. \tag{4.4}$$

Equation 4.4 is used to adjust the vertical dimension of the smallest cell. This

adjusted dimension is denoted as $\hat{y_0}$:

$$\hat{y_0} = \left\lfloor \frac{\alpha - 1}{\alpha^{n+1} - 1} Y \right\rfloor. \tag{4.5}$$

A similar procedure is followed to determine the size of the horizontal dimension of the cells. Let X denote the horizontal dimension of the frame. Starting at the top border of the frame, at position $X/2$, i.e., the mid-section of the frame, the same growth rate α is used to increase the horizontal dimension of cells. Specifically, this dimension is modified in symmetrically from position $X/2$. However, is not adjusted the initial horizontal dimension, as we expect to find most of the changes in objects' size along the vertical dimension. See Appendix A for an example structure construction. The next step is to proceed with feature extraction from the defined STSR.

4.1.3 Feature Extraction

Motion variations influence the descriptiveness of the extracted features. Many motion and change detection algorithms perform well in some types of videos, but most are sensitive to sudden illumination changes, environmental conditions, background/camera motion, and shadows. To this end, substantial efforts have been made by the CDNET initiative to provide a benchmark for testing and ranking existing and new algorithms for change and motion detection [130–133].

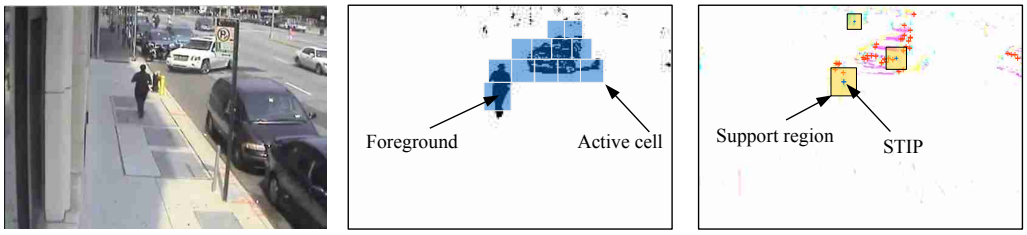
The ADCSF employs motion and change detection algorithms that allow extracting a compact set of very descriptive features. To this end, we use two sources of motion; i.e., background subtraction and optical flow, to extract foreground occupancy and optical flow features, respectively.

Foreground Occupancy Features

Foreground occupancy is remarkably useful to determine presence and long-term events [81]. Foreground occupancy provides the motion source that can help to capture the size and duration of objects in the scene [81]. To this end, we employ a background subtraction method [134].

Applying background subtraction results in a collection of binary masks, one for each frame, where true logical values represent the foreground. We denote these binary masks as B_t for frame t.

We generate a video volume $u \in \mathbb{R}^3$ on B_t for each cell c located at the position (i, j) as described in Section 4.1.2. The video volume u has dimensions $m_x \times m_y \times m_t$, where dimensions m_x and m_y are determined by the horizontal and



a) Input frame b) Features from foreground c) Features fromoptical flow

Figure 4.5: Features extraction. a) Input frame. b) From the foreground mask B_t, we extract features from active cells . c) From absolute frames differences, we detect STIPs using FAST and the corresponding STSRs are encoded using optical flow energy and HOF descriptors. c) Three example support regions of different size.

vertical dimensions of the cell, respectively, and m_t denotes the number of frames which it is fixed for all video volumes.

To calculate the size of the detected objects and their duration in the scene, we count the number of foreground pixels in each video volume u. For each video volume u the feature $F_t \in \mathbb{R}^1$, which represents the foreground occupancy, is calculated as given by the following expression:

$$F_t(i,j) = \frac{1}{N}\sum_{n=1}^{N} u^{(n)}, \tag{4.6}$$

where N is the number of pixels in u. We consider the cells as active if their associated video volumes have a foreground occupancy F_t above a threshold. The cell is active if at least 10% of the pixels in the associated video volume belongs to the foreground (see Figure 4.5b). We further analyse only the video volumes associated with active cells. This exclusion helps to avoid examining regions that mostly depict background, thus reducing frame processing times and false alarms.

Optical Flow Features

To extract features from optical flow, first, we detect STIPs using the FAST detector [21]. FAST is a binary-based technique that detects interest points by comparing the intensity of a particular pixel p with that of its neighbours. If all neighbouring pixel intensities are greater or lesser than the pixel intensity of p, then the pixel is considered to be an interest point. This particular binary-based detector has significant advantages concerning speed over convolution-based ones [98, 103].

In order to discard background information, we apply the FAST detector on the absolute temporal frame differences (see Figure 4.5c). For each spatial location

(x_p, y_p) detected by FAST at the frame difference t_p, we generate a video volume $v \in \mathbb{R}^3$ of size $m_x \times m_y \times m_t$ centred at (x_p, y_p, t_p). The cells' sizes in the structure proposed in Section 4.1.2 provide sizes m_x and m_y, size m_t is constant for all video volumes. The size of the video volume v is given by the spatial location (x_p, y_p) where we detect the FAST. Specifically, m_x and m_y are equal to the horizontal and vertical size, respectively, of the cell in which the space location (x_p, y_p) detected by STIP falls into. We compute the optical flow energy and a HOF descriptor [31] to concatenate the feature $\{O_p(x_p, y_p, t_p), w_p(x_p, y_p, t_p)\}$ for each STIP detected by FAST, where

- $O_p(x_p, y_p, t_p)$ is the optical flow energy computed as:

$$O_p(x_p, y_p, t_p) = \frac{1}{N} \sum_{n=1}^{N} \left\| \left[v_x^{(n)}, v_y^{(n)} \right] \right\|_2, \tag{4.7}$$

  where v_x and v_y correspond to the horizontal and vertical optical flow components, respectively, for the N pixels in video volume v; and

- $w_p(x_p, y_p, t_p)$ is a HOF descriptor; a 5-bin optical flow histogram calculated in the range $[0, 4/5\pi]$ (see Figure 4.4). The histograms are normalised using ℓ_1.

4.1.4 Model Construction

We build an inference model to detect abnormal events. The model, as depicted in Figure 4.1, is composed of four sub-models. Foreground occupancy features and optical flow energy features are analysed separately by two distinct GMMs. A Dictionary Model and a Markov Model process the HOF features.

GMM for Foreground Occupancy

We use the foreground occupancy of the scene to capture variable-sized objects and long-term activity. This motion source allows dealing efficiently with objects that appear for different periods of time in the scene. Foreground occupancy also provides information about the size of objects, which is captured by the number of active cells, as described in Section 4.1.3. We analyse the foreground occupancy of each one (see Equation 4.6) using a GMM (see Equation 4.8) with parameters $\theta^F = \{\pi_k^F, \mu_k^F, \sigma_k^F\}$, representing the weight, mean and standard deviation, respectively, of the kth component of the GMM. The model's elements are determined exhaustively

by iterating the AIC over the model:

$$p_{FG}(F_t(i,j) \mid \theta^F) = \sum_k \pi_k^F \mathcal{N}(F_t(i,j) \mid \mu_k^F, \sigma_k^F), \tag{4.8}$$

where $\mathcal{N}$ is a normal distribution. For the GMM model of Equation 4.8, the AIC compares models in the light of information entropy as a measure of Kullback-Leibler divergence. The AIC for the given model is:

$$AIC(k, F_t) \triangleq log\left(p_{FG}(F_t \mid \theta_{MLE}^F)\right) - dof(k), \tag{4.9}$$

where F_t represents the values to be modelled, whose likelihood is to be maximised by the corresponding distribution of parameters; and θ_{MLE}^F is the set of parameters that result in the MLE. Experimentally, we observe that more than ten degrees of freedom (dof) usually do not provide relevant information. Thus we limit this number to ten.

We also considere the current cell's foreground occupancy, $F_t(i,j)$, to evaluate the likelihood of its immediate neighbouring cells as follows:

$$p_{FGL}(F_t(i,j)) = \prod_{x=i-1}^{i+1} \prod_{y=j-1}^{j+1} \delta_{x-i,y-j} p_{FG}(F_t(x,y) \mid \theta^F), \tag{4.10a}$$

$$\delta_{a,b} = \begin{cases} 1, & a = 0, b = 0 \\ 0.2, & otherwise \end{cases}, \tag{4.10b}$$

where δ is an exception-modified Kronecker delta function.

GMM for Optical Flow Energy

Events like panic and other sudden variations in the scene might not be adequately described by the number of objects in the scene, their size or long-term activity, but rather by the speed of their motion. In order to capture sudden variations in the scene, we therefore use a GMM of optical flow energy with parameters $\theta^O = \{\pi_k^O, \mu_k^O, \sigma_k^O\}$, representing the weight, mean and standard deviation, respectively, of the kth component:

$$p_{OF}(O_p(x_p, y_p, t_p) \mid \theta^O) = \sum_k \pi_k^O \mathcal{N}(O_p(x_p, y_p, t_p) \mid \mu_k^O, \sigma_k^O). \tag{4.11}$$

The model in Equation 4.11 is also estimated by recursively minimising the AIC metric, as done for the GMM of foreground occupancy. The Akaike criterion for

model p_{OF} is then:

$$AIC(k, O_p) \triangleq log\left(p_{OF}(O_p \mid \theta^O_{MLE})\right) - dof(k), \tag{4.12}$$

where O_p represents the values to be modeled and θ^O_{MLE} is the corresponding set of parameters that results in the maximum likelihood estimation.

Dictionary Model for HOF Descriptors

We have to capture the intrinsic activity information of the scene considering that the activity may be very different in different regions. For instance, in a scene depicting a traffic intersection, the activity in the sidewalk may differ a lot from that on the road. However, anomalous events may occur in both areas. Based on this, we propose to create unique dictionaries, one per cell, instead of creating a global dictionary as in [4, 72, 81, 98].

Individual BoF can significantly enhance performance in action recognition, as [32] suggests. For each cell defined, as described in Section 4.1.2, is assigned a dictionary that is generated from the set S of HOF descriptors within the cell. To this end, we use the k-means algorithm to define the cluster centroid $z_i \in \mathbb{R}^5$ in a dictionary, as follows:

$$z_i : \arg\min_{S} \sum_{i=1}^{k} \sum_{w_p \in S_i} \left\| w_p - z_i \right\|_2^2, \tag{4.13}$$

where $w_p \in \mathbb{R}^5$ is a HOF descriptor as defined in Section 4.1.3. We associate the dictionary with a probabilistic vote according to the ℓ_2 distance. The distance d to those *seen* words is $\ell_2 \simeq 0$ if the word is present in the dictionary, and $\ell_2 \gg 0$ otherwise. We calculate the posterior of the distance of the observed words as a normal distribution with parameters $\theta^{DIC} = \{\mu^{DIC}, \sigma^{DIC}\}$, representing the mean and standard deviation of the distribution, respectively:

$$p_{DIC}(d_p \mid \theta^{DIC}) = \mathcal{N}(d_p \mid \mu^{DIC}, \sigma^{DIC}), \tag{4.14}$$

where d_p is the ℓ_2 distance of the word $w_p \in S$ to the cluster centroid z_i.

Markov Model for HOF Descriptors

We employ an FSMC to detect abnormal word ensembles [4]. Markov Models have been successfully used to detect anomalous events in the context of long-term activities [74, 78]. However, these models are usually designed to identify abnormal

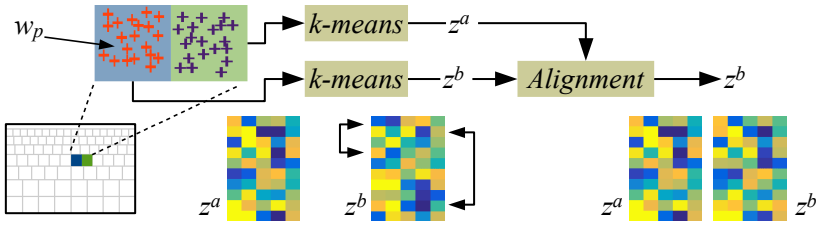


Figure 4.6: Individual dictionaries are built from the set S of HOF descriptors within each cell. The dictionaries are aligned to ensure the correct FSMC transitions.

events in a global context. This context may be difficult to address if the activity in the scene varies significantly across different regions, e.g., the activity in a sidewalk and the road in a scene depicting a traffic intersection. Thus, we use a local model to detect anomalous events by considering the Markov Model of different regions. Let us consider the current state X_l given by the matching label l of the local dictionary; the FSMC probability is then:

$$p_{MRV}(X_{1:L}) = p(X_1) \prod_{l=2}^{L} p(X_L \mid X_{l-1}), \tag{4.15}$$

where L is the number of transitions defined by the total number of labels of the current local dictionary. The matching label index, l, is defined as:

$$l : \arg\min_{l} \left\| w_p - z_l \right\|_2^2, \tag{4.16}$$

and the associated transition matrix, A, is defined as:

$$A_{ij} = p(X_l = j \mid X_{l-1} = i), \tag{4.17a}$$

$$\sum_{j} A_{ij} = 1. \tag{4.17b}$$

We need to know how likely is that words i and j co-occur. The probability of observing the two words $\{i, j\}$ is given by the occurrence n of the words, as follows:

$$A_{ij}(n) = p(X_{l+n} = j \mid X_l = i). \tag{4.18}$$

The order of occurrence of words is not important if the number of analysed frames is limited, as it is in this case. Therefore, we can discard their order of occurrence making matrix A symmetrical.

Since we use a number of different isolated dictionaries, i.e., one per cell, the

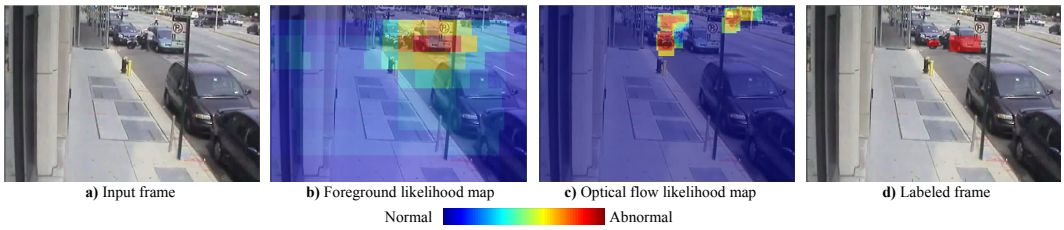


Figure 4.7: Anomaly inference mechanism. From a) incoming frames, the posterior models are evaluated by thresholding the b) foreground occupancy likelihood map and the c) optical flow likelihood map. d) The frame is labelled by evaluating consecutive frames.

FSMC requires that the transition states between adjacent regions correspond to the same matching labels. For example, in the case of two neighbouring cells, a and b, with different labels associated with the same word (see Figure 4.6). Therefore we align a pair of neighbouring dictionaries, $z^a \in \mathbb{R}^{k\times 5}$ and $z^b \in \mathbb{R}^{k\times 5}$, as follows:

$$i : \arg\min_i \left\| z_j^a - z_i^b \right\|_2^2, \tag{4.19a}$$

$$z_j^c = z_i^b, \tag{4.19b}$$

where $z_j^a \in \mathbb{R}^5$ and $z_i^b \in \mathbb{R}^5$ are the words j and i associated with the dictionaries for cells a and b, respectively; and $z^c \in \mathbb{R}^{k\times 5}$ is an empty auxiliary dictionary. By setting z^b equal to z^c, we align the dictionary. After this alignment procedure, the matching labels l_p^a and l_p^b for the word w_p in the dictionary a and b, respectively, are the same, i.e. $l_p^a = l_p^b$. This ensures that the FSMC transition is the same in a local spatial region of the scene.

4.1.5 Inference Prediction

The anomaly inference mechanism works in two joint phases. In the first, the models generate two likelihood binary masks. In the second, these are jointly analysed to determine unusual events.

First phase - mask generation

The model generates two binary masks in this phase. The first, by thresholding the posterior of the foreground occupancy model. Figure 4.7b shows a sample of foreground occupancy likelihood map before thresholding. The second, by thresholding the likelihood of the optical flow energy and HOF descriptor model. Figure 4.7c shows a sample optical flow likelihood map before thresholding. The foreground oc-

cupancy binary mask, $MASK_{FG}$, is then generated by the posterior of each active cell given by the pdf of the model in Equation 4.6, as follows:

$$\gamma_{FG} = -\log\left(p_{FGL}\right), \tag{4.20a}$$

$$MASK_{FG} = \begin{cases} anomalous, & \gamma_{FG} > \epsilon_{FG} \\ normal, & \gamma_{FG} \leqslant \epsilon_{FG} \end{cases}, \tag{4.20b}$$

where ϵ_{FG} is a threshold used to determine if the foreground model vote is normal. Similarly, the posterior of the optical flow energy and HOF descriptors are captured into the binary mask $MASK_{OF}$ as follows:

$$\gamma_{OF} = -\log\left(\prod\left\{p_{OF}, p_{DIC}, p_{MRV}\right\}\right), \tag{4.21a}$$

$$MASK_{OF} = \begin{cases} anomalous, & \gamma_{OF} > \epsilon_{OF} \\ normal, & \gamma_{OF} \leqslant \epsilon_{OF} \end{cases}, \tag{4.21b}$$

where ϵ_{OF} is a threshold used to determine if the optical flow model vote is normal.

Second phase - mask joint analysis

In the second phase, we evaluate the masks $MASK_{FG}$ and $MASK_{OF}$ using a joint criterion. Specifically, if a cell is deemed anomalous in any of the individual masks, then the corresponding frame at time t is marked as anomalous in that region using $MASK_t$, as follows:

$$MASK_t = MASK_{FG,t} \vee MASK_{OF,t}. \tag{4.22}$$

In order to make the inference mechanism more resilient to noise, we use the two consecutive frames at times $\{t, t+1\}$ to determine the abnormality of a frame at time t, as follows [81]:

$$\hat{MASK}_t = MASK_t \wedge MASK_{t+1}. \tag{4.23}$$

The binary mask $\hat{MASK}_t$ then represents the abnormal regions in the current frame as the combination of the models of foreground occupancy and optical flow (see Figure 4.7). Note that one of the advantages of using variable-sized cells is that the anomaly inference mechanism can locate abnormal regions at different levels of spatial granularity according to their position relative to the camera. Figure 4.7 shows an example of a robbery detection with the anomalous areas in red. The

region occupied by the abnormal vehicle is much smaller than those of other vehicles closer to the camera.

Framework Complexity Discussion

The ADCSF generates a limited number of features, thus reducing computational times. Specifically, it creates foreground occupancy features (see Equation 4.6) only for those video volumes whose associated cells are considered as active, thus considerably reducing the number of encoded features of this type. The number of encoded foreground occupancy features may be as low as zero if no activity is present in the scene. This is a significant advantage compared to other methods, e.g., [4, 51], that densely extract features from the entire sequence regardless of the activity in the scene.

We limit the number of generated optical flow features by the number of strongest STIPs detected by FAST. In this thesis, the number of FAST STIPs is the 40 strongest detections. This constraint also considerably reduces the overall number of encoded features of this type. Moreover, we create dictionaries for only those cells where HOF descriptors are found, thus limiting the number of dictionaries to be processed. Section 4.2.2 shows that extracting features is one of the most time-consuming steps.

Regarding the models, when classifying the current frame (see Equations 4.20 and 4.21), the posterior function is evaluated considerably fast due to the linear complexity of the associated normal distribution, $\mathcal{N}$. The FSMC model (see Equation 4.18) is a simple memory access procedure where the label given by dictionary matching gives the matrix index (i, j) in $O(n)$ (see Equation 4.17). Thus, the associated computational complexity is very low.

The generation of the final mask (see Equations 4.22 and 4.23) only involves evaluating two binary masks representing the vote given by the models. Both masks can be evaluated remarkably fast due to their binary nature and the fact that only AND/OR operations are required. Overall, the ADCSF is designed to detect anomalous events in the shortest time possible, making it suitable for online processing. Next Section further confirms this advantage.

Figure 4.8: Abnormal events samples of existing datasets. Example frames of the UMN dataset (first column), which are people running after a clap. The UCSD dataset samples (columns 2 and 3), the unusual events are the presence of non-pedestrian objects. The Subway dataset samples, (fourth column) suspicious events are two people getting out through the subway entrance and the same people entering without payment.

4.2 Experiments

This Section comprises five subsections. In Section 4.2.1 discusses relevant aspects such as sort of events, static characteristic, conditions and goals of the existing datasets. Section 4.2.2 shows the setup of those datasets. Section 4.2.3 describes the metric used to evaluate our method. Section 4.2.4 presents results of the experiments. Finally, Section 4.2.5 comprises a discussion of the results.

4.2.1 Datasets

We use the UMN, USCD and Subway datasets for performance evaluations. used for performance evaluations. In order to evaluate the ADCSF on real video surveillance data, we also use a new collection of realistic videos captured by surveillance cameras with challenging events to be detected. This dataset is hereinafter referred to as the Live Videos dataset (LV dataset). We summarize these datasets next.

UMN dataset

This dataset comprises 11 video sequences depicting people walking in random directions and suddenly simulating panic (see Figure 4.8). Videos are captured in three different scenarios with no camera motion and insignificant illumination changes. Specifically, it comprises two outdoor scenarios with good illumination and one indoor scenario with poor illumination. The videos are captured at 30 FPS. Ground truth of the instants of time when the abnormal events occur is provided; however, no ground truth of the specific abnormal regions is provided.

a) Bank robbery **b)** Stampede **c)** Street robbery **d)** Highway accident

Figure 4.9: Samples of the Live Videos dataset. a) A bank in a robbery, b) stampede at the congregation, c) armed robbery in a street and d) highway accident.

UCSD dataset

This dataset comprises 96 sequences with different crowd densities where the abnormal events correspond to the presence of non-pedestrian entities on a sidewalk (see Figure 4.8). Videos are captured with no changes in illumination or camera motion from two different perspectives overlooking two different sidewalks, resulting in two different scenes: the Peds1 and Peds2 scenes. The ground truth provided allows evaluation at the frame and pixel-levels. For scene Peds1, we use 36 videos for testing and 34 videos for training. For scene Peds2, we use 16 videos for testing and 12 videos for training.

Subway dataset

This dataset comprises two scenes from the entrance and exit of a subway station. Three actors perform unusual activities which include entering without payment, wrong-way direction, loitering and irregular interactions (see Figure 4.8).

Proposed LV dataset

There exist different datasets to develop and test video anomaly detection methods. These datasets usually contain simulated scenes with actors behaving abnormally, e.g. [10, 44, 135]; or more realistic ones with an insufficient number of abnormal events, e.g. [70, 75, 77]. In general, existing datasets present four main drawbacks. First, many sequences have predefined scripts which poorly represent real scenarios captured by surveillance cameras. Second, the available training and test data are often acquired by different cameras or from different scenes. Third, many sequences have ideal environmental conditions which do not represent real scenarios encountered by surveillance cameras. And fourth, many footages usually contain a limited number of abnormal events, or these events are very repetitive.

Based on the previously exposed facts, we propose the LV dataset, which is

a new abundant collection of video sequences captured in challenging conditions by surveillance cameras depicting real abnormal events; e.g., car accidents, robberies, kidnappings, and other dangerous situations.
The LV dataset comprises 28 real sequences of different frame sizes captured at different frame rates in indoors and outdoors scenarios with several illumination changes and some camera motion. All sequences are captured by surveillance cameras (see Figure 4.9 for some examples). The videos depict various crowd densities, from empty scenes to the presence of thousands of people. Anomalous events last from a couple of frames to thousands of frames. The proposed LV dataset comprises video sequences characterised by the following aspects:

- Real events without actors performing predefined scripts with a diverse subject interaction.
- Highly unpredictable abnormal events in different scenes, some of them of very short duration.
- Scenario correspondence, the same scene contains both: the training and test data.
- Challenging environmental conditions; sequences are under changing illumination and camera motion.

Table 4.1 tabulates the main characteristics of this dataset and Figure 4.10 presents examples with detailed explanations.

Table 4.1: Main characteristics of the proposed LV dataset.

Duration	3.93 hours
Frame Rate	7.5 - 30 FPS
Resolution	minimum: QCIF (176×144) maximum: HDTV 720 (1280×720)
Format	MP4 video in H.264
No. of videos/scenes	28/28
URL	https://cvrleyva.wordpress.com/
Anomalous Frames	68989
Events of Interest	34
Abnormal events	Fighting, people clashing, arm robberies, thefts, car accidents, hit and runs, fires, panic, vandalism, kidnapping, homicide, cars in the wrong-way, people falling, loitering, prohibited u-turns and trespassing.
Scenarios	Outdoors/indoors, streets, highways, traffic intersections and public areas.
Crowd density	No subjects to very crowded scenes.

As proposed in [10, 44], LV dataset comprises sequences where the unusual events are those that rarely occur in a scene. In this case, the activity in the scene

a) Wrong Way Sequence: traffic goes in a single direction; suddenly men come out of three cars and people start running; one of them starts shooting (top right corner of the frame) and traffic slows down; motorcycles start circulating in the wrong way.

b) Robbery Sequence: a security guard is at the exit of a supermarket; customers exit the scene after payment; three armed men suddenly brake into the supermarket; they force some customers to lie on the floor and beat one of the cashiers.

c) Traffic Accident Sequence: a two-way road with sidewalks, where pedestrians are walking; a truck crashes into a house hitting a car and a light pole

d) Panic Sequence: video surveillance of customers in a convenience store; suddenly (captured by another surveillance camera) four armed men enter the store and customers start running; some customers try to hide first and later escape through the exit.

Figure 4.10: Example frames of the LV dataset. The boxes highlight the abnormal objects/events.

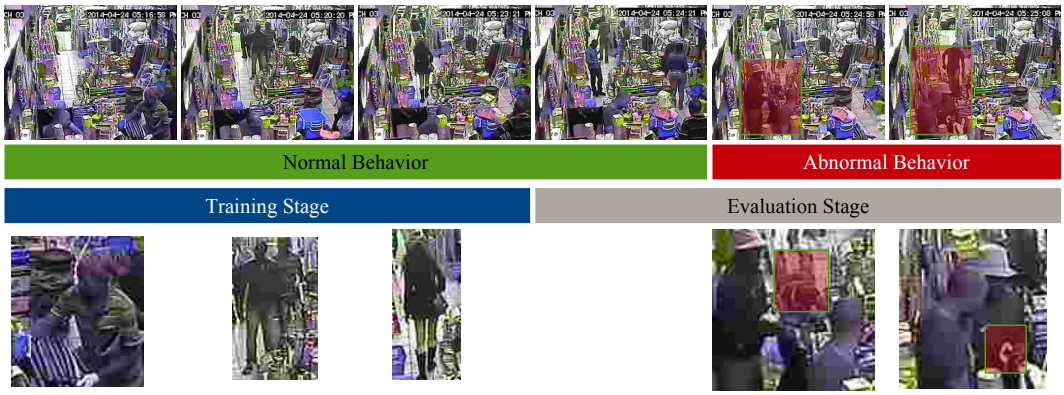


Figure 4.11: Example of normal and abnormal frames from the Robbery sequence of the LV dataset (person on the floor and gun threat). During the training and evaluation stages, normal behaviour is present in the scene (paying, walking, seeing).

may not necessarily increase, for instance, robberies, car accidents and loitering. As proposed in [1, 70, 135], it also includes sequences where the activity in the scene may significantly change during the abnormal event; such as fights, people clashing, fire and panic scenes. Figure 4.11 shows an example, where the cashier next to the point of sale and customers are deemed to be normal. Although the cashier and the customers perform a relatively large number of actions, when the burglars break into the store the activity significantly changes. The burglars beat the cashier, and some customers are forced to lie on the floor.

Some datasets, e.g., the dataset in [77], provide the ground truth for the specific unusual objects. Providing such pixel-level ground truth may be very challenging. As an example, consider the video sequence of Figure 4.11 depicting an armed robbery in a convenience store. Are only the burglar and cashier to be deemed as anomalous for, respectively, showing a weapon and raising their hands? Should the surrounding people witnessing the robbery and trying to escape (or not to interfere) be considered as anomalous too? In this example, we have to define the beginning and end of the chain of individual actions. These actions must then be to correctly determine which objects are abnormal, which may be a difficult task. In the proposed LV dataset, we bound the anomalous event as STSR in the sequence where the main action of interest occurs, e.g., the area where we see the burglar.

All videos of the LV dataset comprise a number of test and training frames. Ground-truth at the ROI-level is provided as a separate sequence of binary masks. No pixel-level ground truth is provided as this type of ground truth is usually very challenging to determine in realistic videos if the abnormal regions contain both

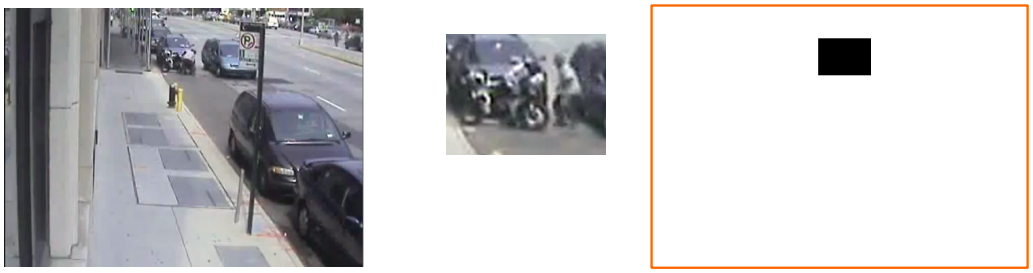


Figure 4.12: Sample frame (left) and provided ROI to detect abnormal events (right). The two burglars putting a motorcycle inside an SUV is the abnormal event, this event is zoomed (centre) for its better appreciation.

foreground and background pixels. As discussed in [77], a method might correctly classify a whole frame as abnormal by incorrectly detecting any region where no abnormal event actually happens. In this case, the system is just lucky as the frame is classified as abnormal without correctly detecting the abnormal event. Evaluations at the ROI-level, using the appropriate ground truth, can therefore avoid this situation.

All sequences in the LV dataset contain both abnormal and normal behaviour, as in the York dataset [44]. Thus, we do not divide the LV dataset into training and test sequences, as in [77]. We provide the labelled frames along with the dataset. As mentioned early, we do not give the ground truth at pixel-level segmentation of unusual objects. Instead, we provide the ROIs in a separate video of the same length as that of the training/testing sequence (see Figure 4.12). A separate file comprises the time stamps with the training frames and those for testing for each sequence.

4.2.2 Experimental Setup

We evaluate the ADCSF against a number of the state-of-the-art video anomaly detection methods [1, 4, 46, 47, 50, 78, 81, 85, 90, 98, 99, 102]. In this work, we consider a method suitable for online processing if the frames meet the FPS rate. For example, for a 30 FPS sequence, frame processing times should be under 33 ms; i.e. (1/30s). Based on this criterion, we see that few methods can attain online performance. Among these, the approach proposed by Lu et al. [47] and that of Biswas and Babu [46] are among the state-of-the-art.

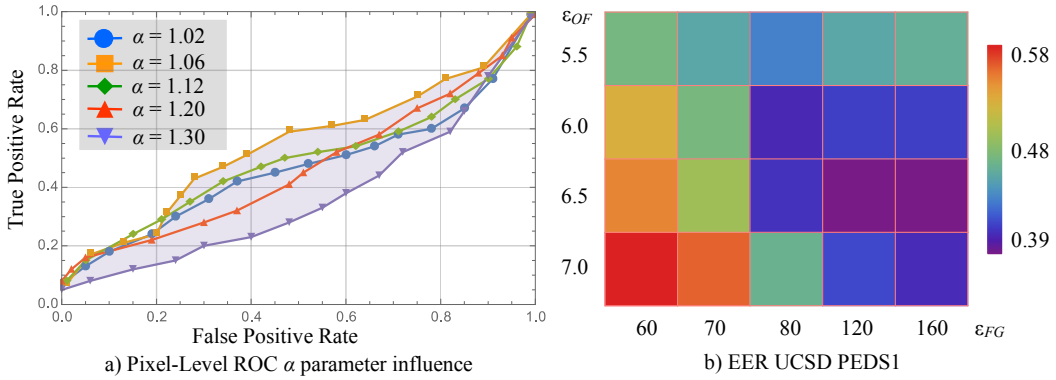


Figure 4.13: Pixel-level EER for scene Peds1 of the UCSD dataset using different values for parameters a) α and b) thresholds ϵ_{FG} and ϵ_{OF} .

ADCSF Parameters

To extract the foreground of the sequences, we use the implementation of Alekhin [136]. For the UMN and UCSD datasets, we record the background from 200 frames using a learning rate of 10^{-2}. During testing, we set the learning rate as 10^{-3}. For the LV dataset, we use the same learning rate parameter of 10^{-2} considering 300 frames. To evaluate the method of Lu et al. [47], we use the code available in [137] with the parameters suggested in the demo code section. To evaluate the method of Biswas and Babu [46], we use the code available in [138] to estimate the interpolation steps; we use a maximum of five GMM model components. We keep the other parameters as proposed by the authors.

For the ADCSF, we empirically determine the value of parameters α, which is the cell growth rate in the proposed cell structure, and ϵ_{FG} and ϵ_{OF}, which are the thresholds in Equation 4.20-4.21. To this end, we evaluate the effect of these parameters on the ADCSF's performance, at the pixel-level, using the Peds1 scene of the UCSD dataset. This particular scene contains relatively small frames with challenging unusual events to be detected. Therefore, we can use this scene to determine the best values for α, ϵ_{FG} and ϵ_{OF} that are appropriate for the other tested datasets.

To determine the value of α, we compute the pixel-level ROC curve using different values of α (see Figure 4.13a). The ROC curve corresponds to the EER over the Area Under the Curve (AUC) of the evaluated method, i.e., EER/AUC, and denotes its precision in terms of its sensitivity. As $\alpha \to 1$, all cells tend to have the same initial size, y_0. If $\alpha \gg 1$, cells tend to increase in size starting from position $(x = X/2, y = 0)$ in the frame, towards $x = 0$, $x = X$ and $y = Y$, where X

and Y denote the vertical and horizontal dimensions of the frame, respectively. The Appendix A.1 comprises details about the cell structure construction. Results in Figure 4.13a shows that the AUC decreases for values of α close to one, i.e., $\alpha = 1.02$. This trend is also evident for large values of α, i.e., $\alpha = 1.3$. For this scene, we can observe that $\alpha = 1.06$ provides the largest AUC. Thus we use $\alpha \in [1.06, 1.2]$ during the experiments.

From Figure 4.13b, we observe that tuning ϵ_{OF} has a more profound impact on pixel-level EER than tuning ϵ_{FG}. Note that ϵ_{FG} values above 80 in conjunction with ϵ_{OF} values above 6.5 attain the lowest pixel-level EERs. Thus we set $\epsilon_{FG} = 80$ and $\epsilon_{OF} = 6.5$ in the experiments to adapt the different characteristics of the evaluated datasets. Table 4.2 summarises the most important parameters of the ADCSF and the recommended range of values we found.

Table 4.2: Most important parameters of the ADCSF.

Parameter	Description	Recommended values
α	Cell size growing rate	1.06 – 1.2
ϵ_{FG}	Anomaly inference foreground occupancy model threshold	80 –125
ϵ_{OF}	Anomaly inference optical flow model threshold	5.5 – 8.5

Some of the sequences in the LV dataset contain large frames, e.g. HD 1200×720. To reduce the complexity, we scale them by sub-sampling to a fixed size of 160×240.

ADCSF Initialisation

We initialise the ADCSF as follows:

1. We build the cell structure according to α and an initial size y_0 (see Appendix A.1). We use this structure to determine the STSRs for the extracted features.

2. Features extraction takes place from the training frames and stored.

3. We process the extracted features to generate the GMMs, dictionaries, and the FSMC.

4. After training, the ADCSF has all models required to start inferring abnormal events.

4.2.3 Performance Metrics

For the UMN, we report the results in terms of the AUC. For this dataset, we use the provided top corner labels as ground truth. Since this dataset offers no pixel-level ground truth, we consider the whole frame as abnormal if at least one region is abnormal. This consideration is the same criterion used in the other evaluated methods. We have to notice that the AUC and EER are similar metrics of a method's performance. Specifically, AUC$\rightarrow$ 1 when EER$\rightarrow$ 0.

For the UCSD dataset, we report results in terms of the EER detection at the frame and pixel-levels. For this dataset, we deem a video frame correctly detected if at least 40% of the pixels are correctly classified [77, 126]. This consideration is the criterion used in all the evaluated methods. The masks provided in [49, 139] comprise the pixel-level ground truth.

For the Subway dataset, we rank the method by counting the number of detected events in each scene. For the LV dataset, we report the results in terms of the ROC curve for the compared methods. Here, we consider the frame correctly identified as abnormal when at least 20% of the ROI is detected. Thus we deem the corresponding event as a true positive.

4.2.4 Results

UMN dataset

Table 4.3 tabulates average AUC values for the UMN dataset. The results compared are as reported in the corresponding cited publication. Note that the ADCSF attains very competitive results compared to no-online (offline) methods, which are expected to outperform online methods. Compared to online methods ours achieves the highest AUC values.

Table 4.3: AUC values for the UMN dataset

Authors	AUC	Frame Processing Time	On-line Performance
Hu *et al.* [102] †	0.977	200 ms	
Li *et al.* [78]	0.996	1100 ms	
Cong *et al.* [87]	0.973	3800 ms	
Zhu *et al.* [90]	0.997	4600 ms	
Lu [47]	0.701	**6 ms**	✓
Biswas and Babu [46]	0.736	14 ms	✓
ADCSF	**0.883**	31 ms	✓

† After optical flow and histogram calculations.

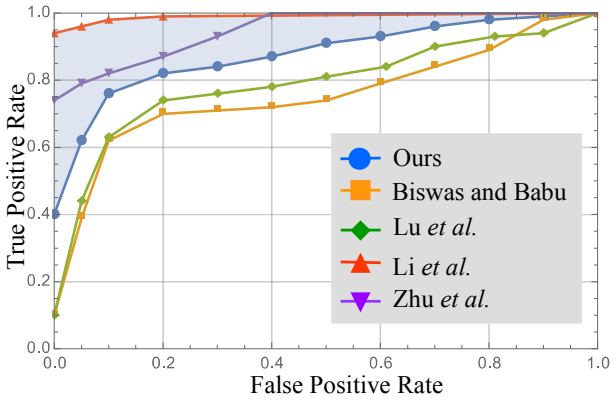


Figure 4.14: ROC curve for the UMN dataset.

We also report the frame-level ROC curve for this dataset. Figure 4.14 shows these results. We can observe that the ADCSF outperforms the online method proposed by Biswas and Babu [46] and that proposed by Lu et al. [47]. Our approach also attains very competitive results compared to other no-online methods optimised for performance. In one case, it outperforms the one proposed by Zhu et al. [90].

UCSD dataset

Tables 4.4 and 4.5 tabulate results for scenes Peds1 and Peds2 of the UCSD dataset, respectively. Results for the other compared methods are as reported in the corresponding cited publication.

Table 4.4: EER for the Peds1 scene of the UCSD dataset.

Authors	EER Frame Level	EER Pixel Level	Frame Processing Time	On-line Performance
Javan and Levine [4]	15	27	190 ms	
Hu *et al.* [102] †	18	36	200 ms	
Cheng *et al.* [98]	19.9	38.8	1100 ms	
Cong *et al.* [87]	23	51.2	3800 ms	
Zhu *et al.* [90]	15	–	4600 ms	
Lu *et al.* [47]	**15**	59.1	**6 ms**	✓
Biswas and Babu [46]	24.66	50.95	14 ms	✓
ADCSF	21.15	**39.7**	31 ms	✓

† After optical flow and histogram calculations.

As expected, no-online methods tend to attain the lowest EER values at both the frame and pixel-levels. However, their frame processing times are considerably longer. For example, the best-reported no-online method, i.e., the approach of [4],

Table 4.5: EER for the Peds2 scene of the UCSD dataset.

Authors	EER Frame Level	EER Pixel Level	Frame Processing Time	On-line Performance
Javan and Levine [4]	13	26	220 ms	
Hu *et al.* [102] †	15	–	200 ms	
Li *et al.* [78]	18.5	–	1100 ms	
Lu *et al.* [47]	22.3	49.8	**6.1 ms**	✓
Biswas and Babu [46]	29.6	42.3	12.5 ms	✓
ADCSF	**19.2**	**36.6**	31 ms	✓

† After optical flow and histogram calculations.

attains a frame processing time six times longer than that achieved by the ADCSF. This very long frame processing time is mainly due to the dense multi-scale sampling used, which is known to be computationally complex. Our method attains a pixel-level EER about 11% lower than that achieved by the online system of Biswas and Babu in [46]. For scene Peds2, the ADCSF achieves better performance at the frame-level than that attained by the online method in [46].

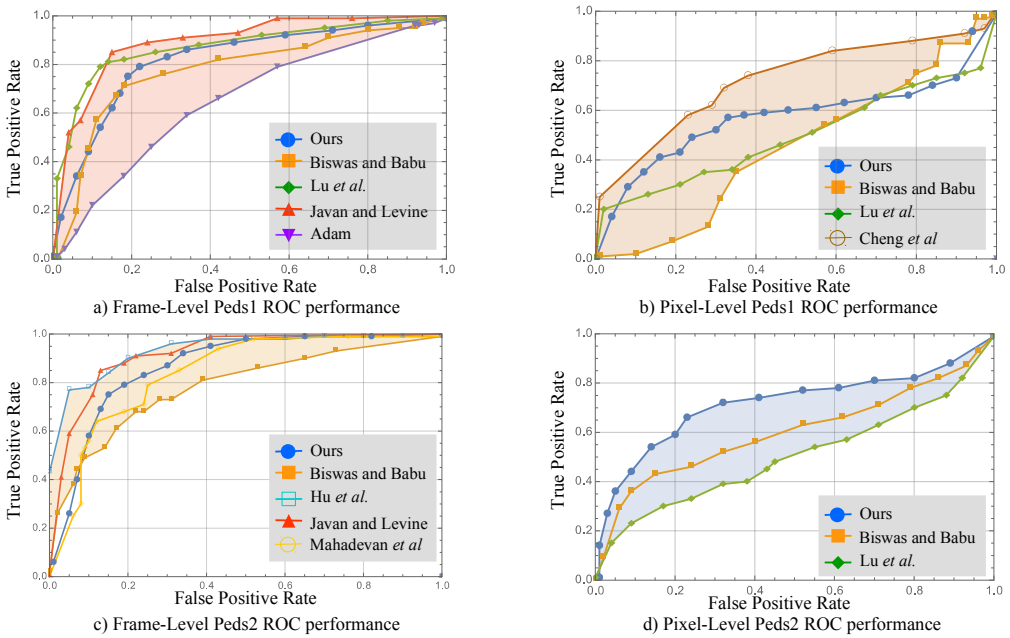


Figure 4.15: ROC Curves for the UCSD dataset at the frame- and pixel-level. (a)-(b) Results for Peds1 scene. (c)-(d) Results for Peds2 scene.

Figure 4.15 shows the ROC curves for the UCSD datasets. From this Figure, one can observe that the ADCSF achieves a very competitive performance compared

to no-online methods. Specifically, at the frame-level, our approach attains results very similar to many of the best performing no-online methods (see Figure 4.15 (a) and (c)).

The ADCSF's results are comparable to methods 10 to 20 times slower, e.g., Cheng *et al.*'s method in [98], which was optimised for performance not for processing times. At the pixel-level, the ADCSF also obtains a competitive performance compared to no-online approaches and significantly outperforms online methods (see Figure 4.15 (b) and (d)). Overall, the ADCSF achieves a very competitive performance on the UCSD dataset compared to no-online methods, while outperforming some of the online ones.

Subway dataset

Table 4.6 tabulates results for the Subway dataset. Results are following the convention for this dataset, i.e., we report the number of detected events by a method for the Entrance/Exit scenes, for each type of anomalous event. The first row indicates the number events to be detected (the ground truth of the dataset). For example, there are 26 Wrong Direction events (WD) from the Entrance scene and 9 from the Exit scene. We indicate this using the notation 26/9.

Table 4.6: Number of events detected for the Entrance/Exit Scene of the Subway dataset for different types of anomalous events: Wrong Direction (WD), No Payment (NP), Loitering (LT), Irregular Interaction (II), Miscellaneous (MISC) and False Alarm (FA).

Authors	WD	NP	LT	II	MISC	FA	On-line Performance
Ground Truth	26/9	13/0	14/3	4/0	9/7	0/0	
Hu *et al.* [102] †	**26/9**	6/0	14/3	4/0	8/7	6/2	
Zhao *et al.* [84]	25/9	9/0	14/3	4/0	**9/7**	5/2	
Biswas and Babu [46]	24/8	5/0	6/2	2/0	5/3	14/10	✓
Lu *et al.* [47]	25/9	7/0	13/3	4/0	8/7	**4/2**	✓
ADCSF	21/6	**9/0**	8/3	2/0	4/2	12/7	✓

† After optical flow and histogram calculations.

From Table 4.6, one can observe that the ADCSF achieves a competitive accuracy compared to other online methods. It notably outperforms other online approaches methods for the No Payment (NP) type of events, i.e., the ADCSF can detect 9 out of the 13 events. We need to notice that these NP events are the most important ones in this dataset, and correctly identifying them is one of the primary motivations behind this dataset. For the Wrong Direction (WD) type of events, the ADCSF also attains a competitive accuracy, very close to the best performing no-online methods, which are designed to achieve high detection accuracy.

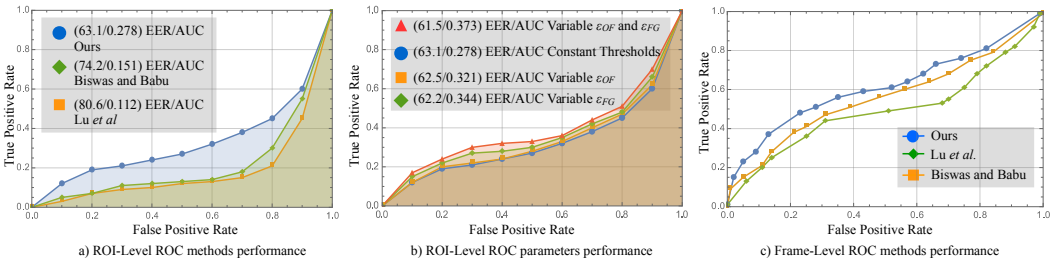


Figure 4.16: ROC curves of compared online methods for the LV dataset. a) We evaluate the ADCSF with constant threshold values ϵ_{FG} and ϵ_{OF}. b) ROC curves of the ADCSF for the LV dataset modifying ϵ_{FG} and ϵ_{OF} values. c) ROC curve LV dataset using a frame-level criterion.

LV dataset

Figure 4.16 shows results for the LV dataset of the ADCSF, the online method of [46] and [47]. We can see that our approach is significantly better than the evaluated online ones. Specifically, the attained EER is nearly 10%-18% lower than that achieved by the online methods in [46, 47].

We have to highlight that the ROC curves in Figure 4.16a) are below the $y = x$ straight line. This characteristic is because true positives are counted only in those cases when a method successfully detects the ROI depicting the abnormal event within a frame. If the system fails to identify this ROI and detects other regions, we count the detection as a false negative. Consequently, this criterion allows us to determine if a method is capable of detecting precisely the area of the scene where unusual events happen. Alternatively, we can label the whole frame as abnormal whenever any region is abnormal, i.e., by following the same frame-level criterion. This manner evidently increases AUC values but prevents measuring if the method is capable of detecting the exact regions that generate the anomaly. Figure 4.16c) shows ROC curves using such a frame-level criterion. We can see that the ROC curves now approach the $y = x$ straight line, as expected. The ADCSF also attains the best performance based on this frame-level criterion.

Table 4.7: Frame processing times and AUC values of online methods for the LV dataset.

Authors	AUC	Frame processing Time
Lu *et al.* [47]	0.112	**6.8 ms**
Biswas and Babu [46]	0.151	13.2 ms
ADCSF	**0.278**	32.5 ms

Table 4.7 tabulates frame processing times and AUC values for the LV

dataset. From this Table, one can observe that the ADCSF attains the highest AUC values and meets online performance for 30FPS videos, which is the highest frame rate in the LV dataset.

Although the other tested online methods are capable of attaining shorter frame processing times for this dataset, we have to notice that their AUC values are close to 50% lower than that obtained by the ADCSF. The shorter frame processing times achieved by [46] and [47] methods are mainly because these methods do not employ optical flow nor background subtraction to collect motion features. They instead use simple temporal gradients and the motion vectors associated with the compressed video sequences. These motion sources consequently decrease frame processing times but sacrifice detection performance.

We also evaluate the effect on the ROC curve of the LV dataset by tuning ϵ_{FG} and ϵ_{OF}. We modify the optical flow model threshold (ϵ_{OF}), while keeping the foreground occupancy model threshold fixed ($\epsilon_{FG} = 6.5$). We test the case of adjusting ϵ_{FG}, while keeping $\epsilon_{OF} = 80$ constant, and also the situation of modifying both. We change these thresholds using the range of values plotted in Figure 4.13. We select the values that provide the highest detection accuracy for each video sequence. Figure 4.16b shows the results of this evaluation. As expected, tailoring both thresholds provides the best performance (see red curve in Figure 4.16b).

An interesting aspect of the previous experiment is that tailoring ϵ_{FG} while keeping ϵ_{OF} fixed provides better performance than tailoring ϵ_{OF} while holding ϵ_{FG} constant (see green curve vs. yellow curve in Figure 4.16b). This outcome is mainly because illumination changes are more drastic than camera motion for the tested dataset. Thus adjusting the ϵ_{FG} threshold has a more direct impact on the ADCSF's performance. This adjusting is a way of specifying how much each model is to be trusted to describe a particular event efficiently.

We observe from the described experiment that thresholds ϵ_{FG} and ϵ_{OF} represent a trust level that indicates how much one can trust the models associated with foreground occupancy and optical flow features, respectively. If one wishes to minimise the effect of a particular model's inference, we set the corresponding threshold to a high value. In this case, we do not trust that specific model, and the overall inference mechanism mostly depends on the other *trusted* model's inference. Therefore, the ADCSF is flexible in this aspect, as it can be adapted according to scene characteristics if these are known *a priori*.

Figure 4.17 shows sample frames showing the anomalous events detected by the ADCSF. Figure 4.18 shows the time proportion of various processes required by the ADCSF during the Detection Stage for a single frame.

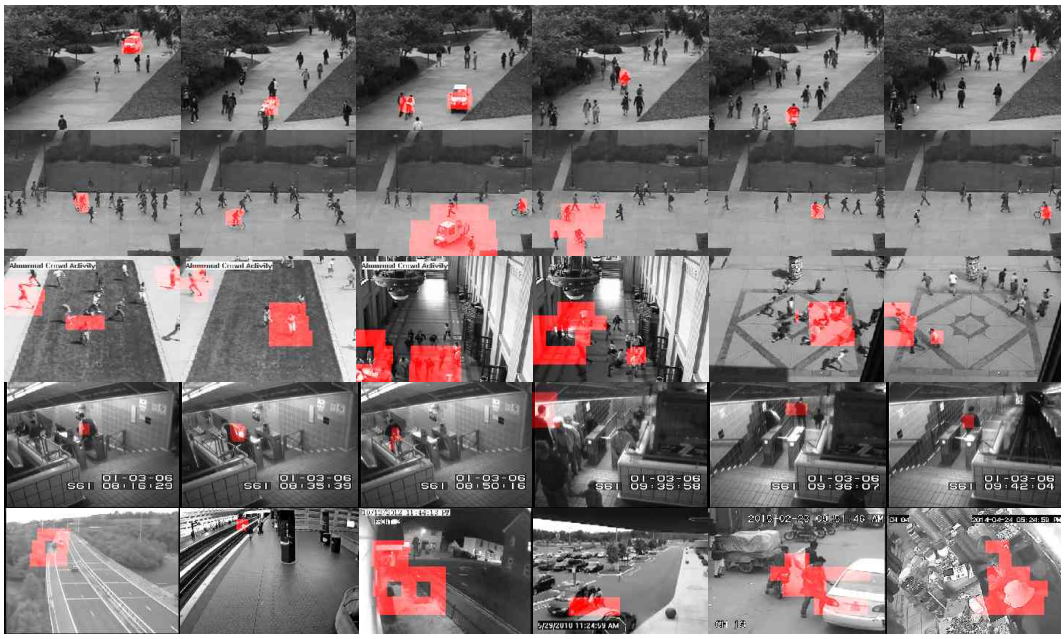


Figure 4.17: Detection samples of the ADCSF. **1st Row.** UCSD Peds1: a man with a trolley, cyclists, small cars and skaters. **2nd Row.** UCSD Peds2: small cars and cyclists. **3th Row.** UMN: people in panic at the moment when they start to run. **4th Row.** Subway: (left to right) Entrance scene showing people entering without payment and walking in the wrong direction. **5th Row.** LV: (from left to right) a lorry hitting a car and capsizing in a highway; a man in a wheelchair falling into the subway tracks; a man destroying private property; a woman being kidnapped outside a shopping mall; an armed robbery; a cashier being beaten by burglars.

4.2.5 Discussions

In the next three subsections, we discuss significant aspects of the ADCSF regarding its efficiency. We finish this Section with essential points that motivate us to address the computational complexity of the feature extraction process. This point serves as the motivation of the next chapter.

Accuracy

Detecting abnormal events in real scenes is challenging. However, the ADCSF is capable of detecting them outperforming other online methods.

For example, let us take the frame in row 5, column 1 of Figure 4.17, which shows a car accident. In this sequence, the ADCSF can detect most of the video frames comprising the accident. The evaluated online methods in [46, 47] are not able to identify this event. The main reason for the poor performance of these two other online methods in this sequence is the fact that moving objects tend to

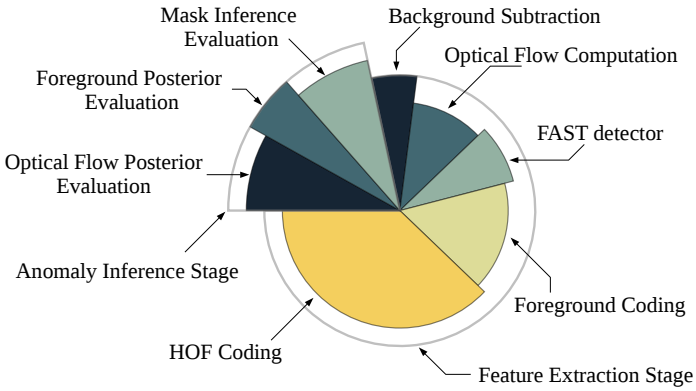


Figure 4.18: Required time by various processes of the ADCSF during the detection stage for a single frame.

slow down when the abnormal event occurs. Consequently, the frame differences, which are the core of both methods, cannot provide features from the region where the accident takes place. This sequence is also an excellent example to showcase the advantages of the variable-sized cell structure, where the small cells capture the event. Therefore, the ADCSF can accurately detect the ROI depicting the car accident.

Another example that demonstrates the advantages of our proposed cell structure is the frame in row 5, column 2 of Figure 4.17, which depicts a man in a wheelchair falling into the subway tracks. In this case, the region where this unusual event takes place is very far from the camera. Thus the ROI to be detected is very small. The proposed coarse-to-fine cells help to accurately identify this event since the method creates enough features from that region at the correct size.

The two other evaluated online methods also fail to identify this particular accident. Let us take now the frame in row 5, column 3 of Figure 4.17. In this case, the abnormal event corresponds to a man breaking into private property and causing some damage to it. The scene has poor illumination, and consequently features based on STIPs are expected to perform poorly. For this scene, the ADCSF profits from the fact that two sources of features are available; those extracted foreground and those from optical flow. Even if those features from optical flow are not descriptive enough, those from foreground occupancy help the ADCSF to detect this event correctly. For this particular scene, the other evaluated online methods fail to recognise this ROI, and instead, they incorrectly identify other regions as abnormal. This discussion plainly shows the advantages of the ADCSF.

Time performance

We implemented the ADCSF in MATLAB and tested on a 2.7GHz CPU with 8GB of RAM. We do not parallelise the code, and we do not use GPU arrays to speed up the computations. One can see, from Figure 4.18, that encoding HOF descriptors are the most expensive step. This aspect is mainly due to the fact that the ADCSF has to calculate every orientation of each pixel in every STSR detected by FAST.

Note that the processing times of the likelihood modelling are much lower than those of the feature extraction process. This benefit is because the ADCSF only extracts features for a limited number of support regions and strongest detected FAST STIPs. This limitation significantly reduces the total number of features to be encoded/processed and is the primary aspect of the ADCSF that helps to reduce overall computational times.

Feature Extraction

From Figure 4.18, we can see that feature extraction is the most time-consuming step. Logically, we need to improve accuracy keeping in mind computational times. The HOF encoding is high computational costly limiting us to explore other well-known computer vision techniques still achieving online performance. This drawback is mainly because feature extraction requires approximately one-third of the whole processing time. This aspect motivates us to propose new features and is the basis of the next chapter.

4.3 Conclusion

This chapter proposes an online framework for video anomaly detection. The ADCSF extracts a compact set of features from foreground occupancy and optical flow. The ADCSF employs a novel variable-sized cell structure which allows extracting features from a limited number of different support regions in a fine-to-coarse fashion. This helps to process a significantly smaller number of features than those processed by dense-scanning based methods.

We evaluated the ADCSF on the popular UMN, UCSD and Subway datasets, as well on the LV dataset, which is a new collection of realistic sequences captured by surveillance cameras under challenging environmental conditions. During the evaluation, we observed that there usually is a trade-off between computational times and detection accuracy. However, the ADCSF manages to attain high detection accuracies while achieving online performance thanks to the compact set of features and

the models used to efficiently process them. Specifically, the ADCSF outperforms online methods, while being very competitive among no-online methods.

As part of the evaluation, we also showed that the ADCSF is flexible to be tailored to the characteristics of the sequences, if these are known a priori, in order to improve performance. Our future work is aimed at further enhancing the ADCSF's detection accuracy by exploiting this flexibility; specifically, by considering the optimization of the ADCSF's parameters given a particular set of environmental conditions used to capture a sequence.

This page intentionally left blank.

Chapter 5

Action Recogntion using Binary Features

This chapter presents our contributions in the field of visual features for video analysis tested on action action recognition. Although there is a vast work in this computer vision task, still there is a lack of visual features aimed to achieve very short processing times. As exposed in Section 2.3.1, the related work is based on capturing information via the pixels orientation coming from many motion sources. In the specific case of action recognition, orientation-based video descriptors demonstrate outstanding performance regarding accuracy. However, their computational complexity is not appropriate for fast video processing, as Section 2.3.3 discusses. This aspect leads us to propose efficient descriptors for video analysis.

Contributions: This Chapter exploits the fact that binary descriptors are well-known to reduce significantly computational times and storage requirements [115, 116]. As stated in Section 2.3.2, the principles of some of the existing work of 2D binary descriptors – e.g., patterns, best-pair seeking – not easily extrapolate to the spatio-temporal domain with competitive results regarding accuracy[1]. Based on these facts, our contributions in this chapter are:

- Binary features **designed specifically** for the spatio-temporal domain that achieve competitive accuracy.
- The new 3D patterns of the 3 Dimensional Binary Pair Differences (3DBPD) and Binary Wavelet Differences (BWD) descriptors proposed to create binary features.

[1]Henceforward, in this chapter, we refer to accuracy as the Correct Classification Rate (CCR) achieved using a particular feature in a framework which correctly classifies one action among an N number of actions.

- We **binarise** the optical flow's displacements to create a binary feature. We name this techinque Binary Dense Trajectories (BDT). We also present its improved version Binary Improved Dense Trajectories (BIDT), extracted from homographies and individual motion components.

- A FVs variant for binary data, which is an alternative feature representation to BoF.

The rest of this chapter is organised as follows: In Section 5.1, we describe the proposed binary-based descriptors 3DBPD, BWD, BDT and BIDT. In Section 5.2, we present experiments and discussions using the descriptors in a BoF+SVM pipeline. In Section 5.3, we present experiments and discussions using the proposed FV. We conclude this chapter in Section 5.4.

5.1 Proposed Binary Descriptors

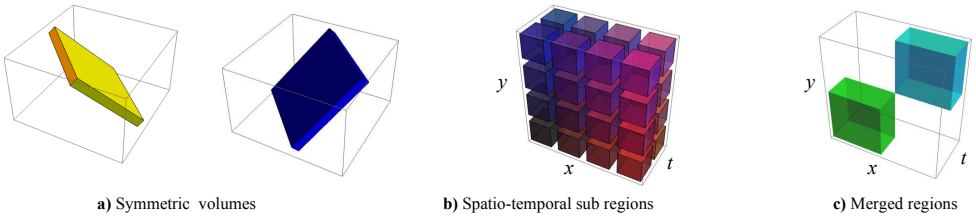
The proposed descriptors are a binary vectorisation of the STSR. To detect the STSR, we use the STIP and the Dense Trajectory (DT) detectors. The 3DBPD and BWD descriptors are calculated based on STIPs, while the BDT and BIDT descriptors are calculated based on DTs.

The 3DBPD and BWD require defining a STSR around the STIP. We use the same criterion for both. First, we detect the STSR using the Laptev and Lindenberg detector [2]. This detector retrieves the isotropic response of a second order Hessian, whose integration scales over the spatial and temporal domain are σ_i^2 and τ_i^2, respectively. The video volume v_i of size $n_x = n_y = \lfloor 9 \cdot \sigma_i \rfloor$ and $n_t = \lfloor 9 \cdot \tau_i \rfloor$ is the STSR around the STIP, whose size corresponds to nine times the scale at which the STIP is detected.

Both descriptors require to estimate their orientation as [115] suggest; we use the same estimator for both. We proposed to determine the dominant orientation using the *intensity centroid* estimator proposed by [117] to make the 3DBPD and BWD descriptors rotationally invariant.

The potential problem of rotating a video volume is that the symmetric volumes may have the same centroid despite their orientation. In this case, the vector that represents the dominant orientation is very sensitive to noise due to its small size. To illustrate this effect, Figure 5.1a) shows two different synthetic volumes with the same centroid. For this case, the intensity centroid estimator

[2] We compare BWD with the descriptors proposed in [6, 106]; thus we use the same detector. Other detectors might not provide a fair comparison to rank the descriptor.



a) Symmetric volumes **b)** Spatio-temporal sub regions **c)** Merged regions

Figure 5.1: **a)** Two symmetric volumes with same centroid positions. **b)** The spatio-temporal grid used to define sub-regions within a video volume. **c)** An example of two regions created by merging sub-regions as defined by the proposed grid.

does not represent the orientation accurately whenever the video volumes tend to be symmetric around the geometric centre of the temporal dimension.

To overcome this problem, we determine the orientation by splitting the video volume into two along the temporal dimension. The centroid is then calculated using the first half of the video volume (i.e., the first half according to the temporal dimension). The intensity centroid estimator consists of determining the dominant orientation using the moment of the video volume v_i. To discard background information, we calculate the volume's temporal differences. Next, we estimate the centroid as:

$$m_{pq} = \sum_{1 \leqslant x \leqslant n_x} \sum_{1 \leqslant y \leqslant n_y} \sum_{1 \leqslant t \leqslant n_t/2} x^p y^q v_i(x, y, t), \tag{5.1a}$$

$$c = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right). \tag{5.1b}$$

We set the origin o at the geometrical centre $(n_x/2, n_y/2)$ and estimate the volume orientation using the angle of $r = \overline{oc}$:

$$\theta = \operatorname{atan2}\left(r_y, r_x\right). \tag{5.2}$$

We bin the angle θ into four quadrant directions, i.e. $\theta \in [0, 3\pi/2]$, and rotate the video volume using the flip operation as follows:

$$v_i(x, y, t) = \begin{cases} v_i(x, n_y - y, t), & \theta \simeq \pi/2, \\ v_i(n_x - x, y, t), & \theta \simeq 0. \end{cases} \tag{5.3}$$

The video volume is rotated using the flip operation instead of matrix multiplication as in [11]. We use this operation because rotating a video volume without a corresponding interpolation step usually involves a significant loss of data. Ex-

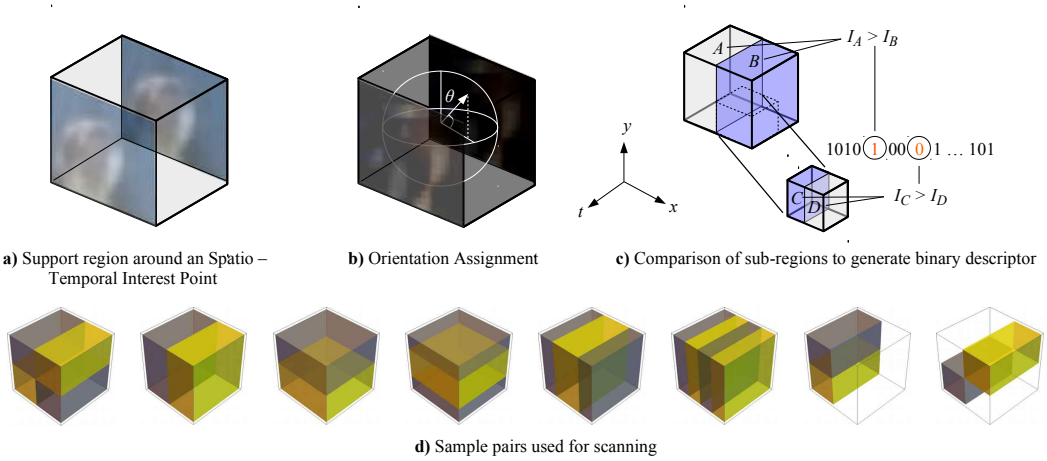


Figure 5.2: The 3DBPD descriptor. **(a)** Video volumes are defined around STIPs. **(b)** Estimation of the dominant orientation of video volumes. **(c)** Comparisons of the mean values of the spatio-temporal sub-regions generate a binary string (sub-section 5.1.1). **(d)** Sample pairs of sub-regions defined with a video volume. Each sub-region is depicted in a different color; yellow and gray. Pixel locations depicted in the same color belong to the same sub-region.

perimentally, we observe that such procedure has two significant impacts: the descriptor's accuracy is decreased and its computational time is considerably high due to the $O(n^3)$ complexity of the interpolation. After determining the video volume orientation, we proceed to extract a binary feature from it.

5.1.1 3D Binary Pair Differences

Similar to the 2D patterns proposed by [114, 116, 140], we define sub-regions of the STSR to capture motion information of video volumes. To discard the background – which generally does not convey motion information – we define the sub-regions after computing the gradient along the time dimension of video volumes.

We consider that comparing pairs of sub-regions conveys more descriptive information than comparing individually isolated ones. Moreover, we also expect that analysing different pairs of sub-regions following a specific pattern would provide more descriptive information than following a random pattern [18]. Following a specific pattern also helps to avoid further finding the best pairs of sub-regions, as in [116]. Based on these observations, we analyse different pairs of sub-regions following a specific pattern. This analysis follows a principle based on LBP, which consists of comparing the values within two regions. From this analysis, we extract a binary features.

We define 32 sub-regions distributed symmetrically with respect to the STIP. The regions result from merging the $4 \times 4 \times 2$ regions of Figure 5.1b into two larger regions as Figure 5.1c illustrates. Figure 5.2 shows sample pairs of these sub-regions, where pixel locations in each of the two different colours shown belong to one of the two sub-regions. Each of the 32 different pairs is used to generate one logical value. Specifically, we compare the mean values of the two sub-regions r_1 and r_2 in a pair P_m as follows:

$$C(P_m) = \begin{cases} 1, & \text{if } \sum v_i(P_m^{r_1}) > \sum v_i(P_m^{r_2}) \\ 0, & \text{else} \end{cases}, \tag{5.4}$$

where $\sum v_i(P_m^{r_n})$ represents the sum of all the absolute temporal difference values within the region r_n of P_m. The binary feature F_i is the concatenation of all logical values calculated for all 32 pairs of sub-regions:

$$F_i = \sum_{0 \leqslant m < M} 2^m C(P_m), \tag{5.5}$$

where M the number of compared pairs. Further analysing smaller sub-regions is also discriminating [140]. Then the next step is to divide the video volume into $2 \times 2 \times 2$ sub-volumes. We also define pairs of sub-regions within each sub-volume. These have four different configurations and correspond to first four pairs illustrated in Figure 5.2d (left to right). The binary generation of the sub-regions is as Figure 5.2c illustrates.

The final binary descriptor is the concatenation of the 32-bit string obtained by analysing the 32 different sub-regions plus the eight 4-bit strings obtained by the eight sub-volumes. This process leads to a binary vector with a dimension of $32 + 8 * 4 = 64$.

5.1.2 Binary Wavelets Differences

The proposed BWD descriptor has the same motivation as the 3DBPD descriptor (computational time and storage requirements) and further enhance accuracy. Examining 3DBPD, we notice three flaws.

(I) As dividing and scanning the small regions, bits are more randomly generated and consequently hinder accuracy. (II) 3DBPD generates bits given importance only to the spatial domain. The patterns always capture the symmetry in the spatial domain giving no relevance to the symmetry in the temporal domain. (III) By dividing the volume into small sub-volumes more computations are required.

We have to overcome the explained drawbacks. To this end, we propose to

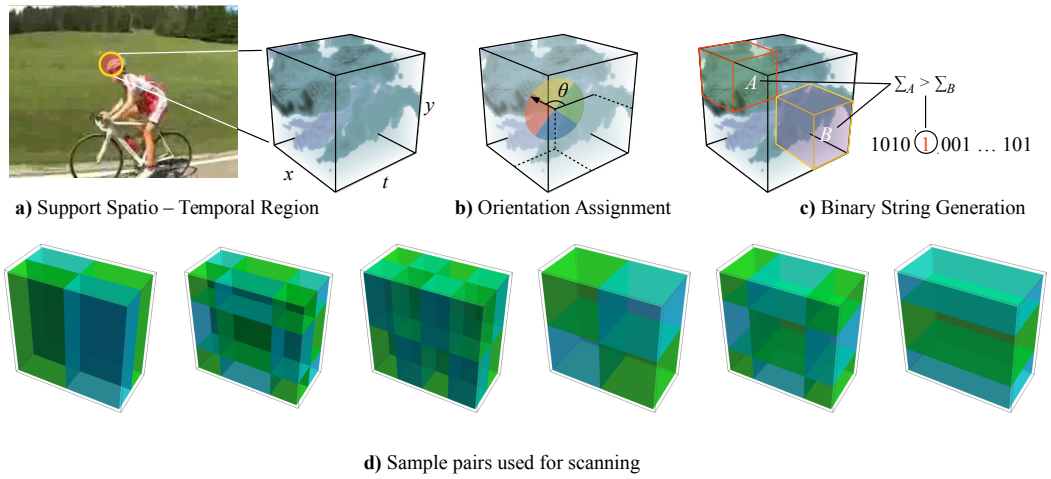


Figure 5.3: Outline of the BWD descriptor computation. **a)** We select the STSR and calculate its absolute temporal differences. **b)** We determine the dominant orientation and rotate the video volume. **c)** We compare different pairs of regions within the video volume to generate a binary feature. **d)** Sample pairs used to generate the binary string.

generate bits only from large regions and avoid further scanning the small ones. However (as Section 5.1.1 shows), we create 32 bits from the small regions; thus we have to extract 32 more bits. Experimentally we can show that 32 bits are not descriptive enough as binary features. To address this problem, we propose temporal-symmetric patterns to improve accuracy and create 32 more bits.

Temporal-symmetric patterns (e.g. first three patterns left to right in Figure 5.3d) capture essential information of those video volumes with sudden motion changes. We tested two type of patterns for the BWD: using 64 patterns (I) keeping spatial symmetry, and (II) keeping temporal symmetry. The results are roughly equal. However, when we decided to combine 32 spatial-symmetric + 32 temporal-symmetric, the descriptor boosted its accuracy. The best-obtained accuracy is when combining spatial-symmetric and temporal-symmetric patterns.

Regarding spatio-temporal symmetry, we can conclude that when gathering information to create a binary descriptor, *it is equally important what the spatial-symmetric and temporal-symmetric patterns capture*. Figure 5.4 illustrates this advanced concept, where the time-symmetric and spatial-symmetric regions entirely comprise the zebra moving parts.

The BWD descriptor captures slow term motion with the spatial-symmetric patterns and fast motion with the time-symmetric patterns, making the descriptor more accurate. This combination is the main contribution of the proposed BWD descriptor compared with the 3DBPD descriptor.

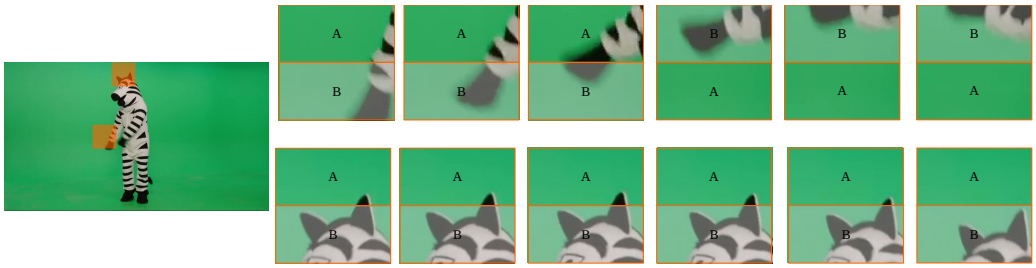


Figure 5.4: (top) Leg and (bottom) head sequence. For both sequences, the region B captures most of the moving parts while A the background. In the top sequence the temporal symmetry is more important than the spatial. In the bottom sequence, the spatial symmetry – on the contrary – is more important.

The BWD descriptor then compares the values of the two regions in each pair to produce an M-bit string, where each bit represents the result of each of the M comparisons. Figure 5.5 depicts the $M = 64$ patterns used to define the pairs of regions in this thesis.

Note from Figure 5.5 that each pattern indeed divides the video volume into two regions, r_1 and r_2. These are symmetrical in either space or time. For example, patterns (1)-(3) in this figure are spatial-symmetric while patterns (2)-(4) are temporal-symmetric. The difference between patterns (3)-(4) and (1)-(2) is that the latter contain void regions that are not considered during encoding. Patterns with void regions allow reducing overlapping in the spatio-temporal regions being analysed, which helps to produce descriptive binary strings. We compute temporal-symmetric patterns as the complement to spatial-symmetric patterns in the range $[0, t/2]$; e.g., (1) and (2), where we say that (2) is complementary to (1). A pattern complementary to a symmetric one is therefore always temporal-symmetric and generated from a spatial-symmetric pattern.

We generate a logical value by comparing the two regions r_1 and r_2 using Equations 5.4 and 5.5. To this end, we employ the BWD patterns (see Figure 5.5) without dividing the video volume v_i as the 3DBPD descriptor needs. Therefore, we generate a 64-bit binary string (feature) after this encoding. The proposed BWD is this M-dimensional binary vector, $\mathbb{R}^3 \rightarrow \mathbb{R}^M$. In other words, we encode each video volume v_i into an 8-byte string F_i. Section 5.2.3 shows the benefits of this compact coding in terms of computational time and storage requirements.

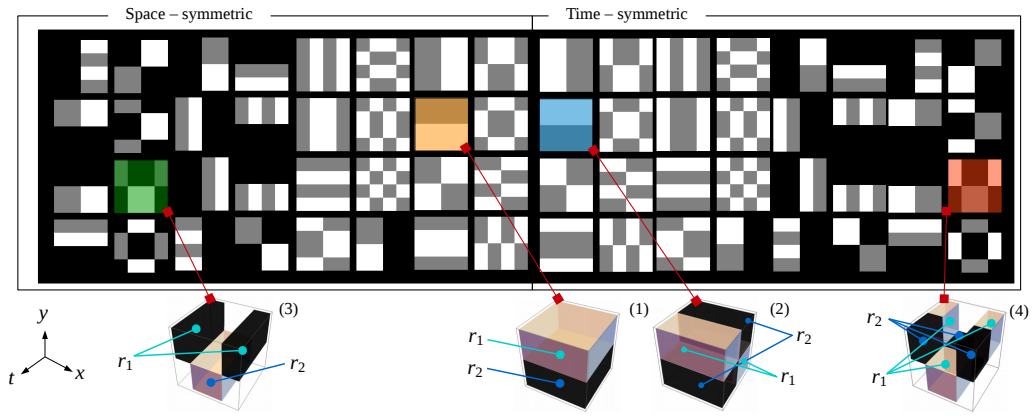


Figure 5.5: The BWD patterns. Each pattern defines the two regions, r_1 and r_2, to divide the video volume. We illustrate four sample patterns: (1) is complementary to (2), while (3) is complementary to (4).

5.1.3 Binary Dense Trajectories

Although STIP-based descriptors have acceptable accuracy in action recognition, the DT detector and its associated descriptor are far superior. Thus, we propose a binary-based descriptor based on DT. DT is an efficient alternative to STIP detectors because it is robust to camera motion. Hence, the motivation of capturing the optical flow into a binary feature. Additionally, we have to encode the STSR provided by the DT detector. To this end, we can use the BWD descriptor.

The main obstacle to generate a binary descriptor based on DT is that it provides a feature comes in an incompatible format, i.e. double data precision. We explored to concatenate the BWD descriptor with the quantisation of the DT descriptor. The results of this concatenation in terms of accuracy are very poor. Therefore, it is necessary to propose a more elaborate formulation. To this end, we propose the BDT descriptor.

The BDT descriptor works as follows. From non-homogeneous regions, we track points using dense optical flow (Figure 5.6a). To this end, we use the detector of [7] to obtain the trajectories from these non-homogeneous regions. We map a trajectory T_t into its binary representation by binning the constituent displacements into six directions according to their normalised magnitude (Figure 5.6b). Note that we need to take into account the magnitude of the constituent displacements of T_t before binning their direction. Short displacements may generate similar binary strings making the descriptor very sensitive to noise. Large displacement, e.g., due to camera motion, can also present the same problem.

To tackle the previous problems we normalise the displacements of T_t with

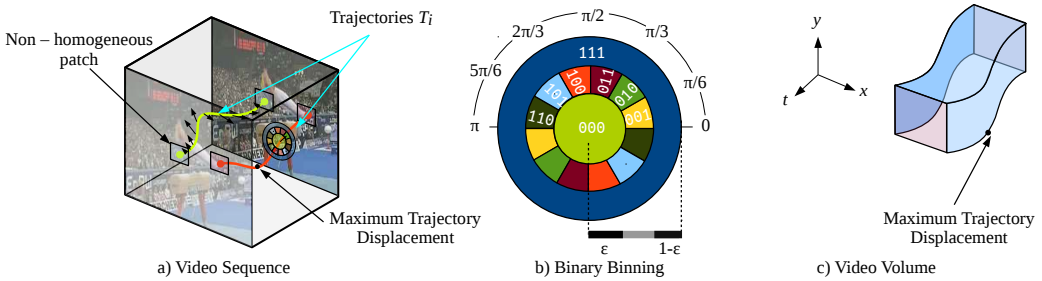


Figure 5.6: Outline to calculate the BDT descriptor. **a)** From non-homogeneous regions, we track points. **b)** We bin the displacements associated with a trajectory into oriented-magnitude directions. **c)** Trajectories describe irregular volumetric regions (video volumes). We concatenate the binarised trajectory with the BWD descriptor obtained from the irregular video volume.

respect to the largest displacement. Next, we bin each constituent displacement if its normalised magnitude is within the range $[\epsilon, 1-\epsilon]$. This range represents the magnitude of those displacements that capture the most stable orientations. We normalise the L constituent displacements of T_t, starting at time t, as follows:

$$\hat{T}_t = \frac{(\Delta p_t, ..., \Delta p_{t+L-1})}{\texttt{max}\|\Delta p\|}, \tag{5.6}$$

where p_t is the points tracked by dense optical flow at time t and Δp_t represents the associated displacement before normalisation. The associated displacement after normalisation is denoted by $\Delta\hat{p}_t$

We represent those displacements with a normalised magnitude less than ϵ (regardless of their direction) using the three bits 000. For those displacements with a normalised magnitude greater than $1-\epsilon$ (regardless of their direction) we use bits 111. We bin those displacements with a normalised magnitude within the range $[\epsilon, 1-\epsilon]$ into one of the orientation bins, each represented by three bits. To make the trajectory directional-invariant, these bins have a π period. We bin those displacements as follows:

$$B(\Delta\hat{p}_t) = \begin{cases} k : \texttt{argmin}_k(\| \Delta\hat{p}_t - b_k \|), & \text{if } |\Delta\hat{p}_t| \in (\epsilon, 1-\epsilon) \\ 111, & \text{if } |\Delta\hat{p}_t| \geq 1-\epsilon \\ 000, & \text{if } |\Delta\hat{p}_t| \leq \epsilon \end{cases}, \tag{5.7}$$

where b_k is the unitary vector representing the kth ($k \in [2, 6]$) bin with a direction in the range $[0, \pi]$. Figure 5.6b illustrates the 3-bit values used to represent the normalised displacements. Note that we create an irregular video volume v_i from

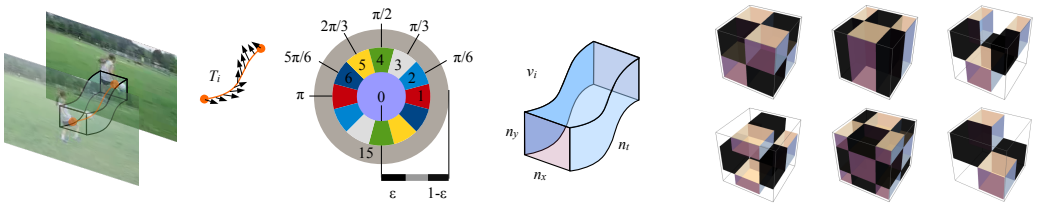


Figure 5.7: BIDT feature extraction. **a)** From the video sequence the trajectory defines the video volume to be encoded. **b)** We encode this trajectory using a binary-scheme according to the displacements' directions and magnitudes. **c)** We encode the generated video volume using the **d)** BWD descriptor.

the tracked points (Figure 5.6c). The binary vector G_i representing trajectory $T_{t,i}$ associated with v_i is then formed by the concatenation of the L binned normalised displacements as:

$$G_i = \sum_{0 \leqslant t < L} 2^{3t} B(\Delta \hat{p}_t), \tag{5.8}$$

where $B(\Delta \hat{p}_t)$ is the trajectory binarisation for an L-length trajectory $T_{t,i}$. The string G_i has thus a dimension $3L$.

We use the irregular video volume v_i associated with trajectory $T_{t,i}$ to compute the BWD descriptor using Equation 5.5. The BDT descriptor is then the binary vector H_i, which comprises the concatenation of F_i and G_i:

$$H_i = F_i +\!\!+ G_i. \tag{5.9}$$

Vector H_i has dimensions of $N + 3L$. The BDT descriptor captures in one single binary vector essential motion information of the dense optical flow as well as that obtained by comparing various spatio-temporal regions.

5.1.4 Binary Improved Dense Trajectories

Improved Dense Trajectories [141] (IDT) is an upgraded version of DT. Thus we decide to use this detector to extract trajectories from the video sequence. As DT, from non-homogeneous regions, points are tracked using dense optical flow. However, in order to make it robust to camera motion, we employ homographies between consecutive frames. Human movement can make this task difficult to achieve because does not belong to the camera motion. Thus, we discard the optical flow coming from human regions. After camera motion estimation, we require the tracked points to generate STSRs around them. These STSRs define a video volume to be encoded.

To encode the tracked points into a binary format, we use an orientation

invariant binning of the trajectory as in Section 5.1.3. In this case, we slightly modified the binary binning scheme by adding more bits to the short and long displacement (see Figure 5.7b).

For a 4-bit binning scheme, we represent those displacements with normalised magnitude less than ϵ using $k = 0$ or string 0000. Displacements with normalised magnitude greater than $1 - \epsilon$ with $k = 15$, i.e., 1111. If the displacement magnitude is within the range $[\epsilon, 1 - \epsilon]$, we then bin into one of the k orientation bins. In summary, we bin the displacements as follows:

$$B(\Delta\hat{p}_t) = \begin{cases} k : \arg\min_k \parallel \Delta\hat{p}_t - b_k \parallel & : |\Delta\hat{p}_t| \in (\epsilon, 1-\epsilon) \\ 1111 & : |\Delta\hat{p}_t| \geq 1 - \epsilon \\ 0000 & : |\Delta\hat{p}_t| \leq \epsilon \end{cases}, \tag{5.10}$$

where b_k is the unitary vector that represents the kth bin with a direction in the range $[0, \pi]$ ($k \in [1, 6]$). Figure 5.7b illustrates the binning process to encode the trajectories. We form the binary vector F_i representing trajectory T_i by the concatenation of the L binned normalised displacements as:

$$F_i = \sum_{0 \leqslant t < L} 2^{4t} B(\Delta\hat{p}_t), \tag{5.11}$$

where $B(\Delta\hat{p}_t)$ is the trajectory binarization of the L-length trajectory T_i. The string F_i has a dimension $4L$.

We convolve the irregular video volume v_i associated with the trajectory T_i (see Figure 5.7b) with a temporal difference operator $d_t = [-1, 1]$ to discard background information. At this point, we generate two video volumes. As [31] suggest, by analysing separately horizontal and vertical components significantly improves accuracy in action recognition. Thus, we propose to capture horizontal and vertical motion information.

From the video volume v_i, we compute its horizontal v_i^x and vertical v_i^y motion components using the Prewitt convolution operators $d_x = [-1, 0, 1]$ and $d_y = [-1, 0, 1]^\top$:

$$v_i^x = d_x * (d_t * v_i) \tag{5.12a}$$

$$v_i^y = d_y * (d_t * v_i). \tag{5.12b}$$

After calculating the components v_i^x and v_i^y, we estimate their orientations. Then we extract a binary feature vector from the video volumes v_i^x and v_i^y. To extract a

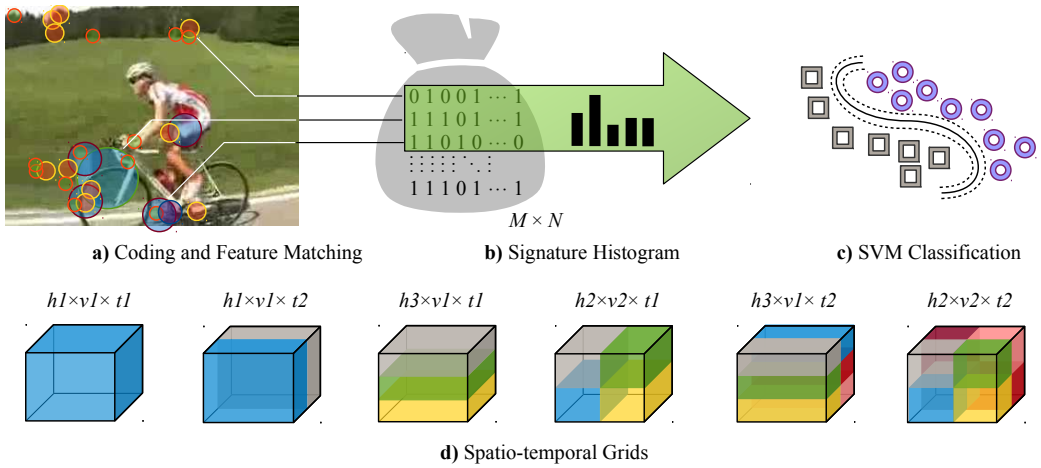


Figure 5.8: Action Recognition Pipeline. **a)** Spatio-temporal regions are encoded using the BWD or BDT descriptors and matched **b)** to the M-words dictionary (BoF). Signature histograms are generated and **c)** an SVM classifier labels the test videos. **d)** An example of the spatio-temporal cell used to divide the video sequence into h(horizontal), v(vertical) and t(temporal) cells. Each cell represents a channel.

binary string from the irregular video volume, we use the 64 pairs of Section 5.1.2. The final descriptor is the binary vector H_i, that results from the concatenation of the trajectory encoding (Equation 5.11) and two BWD (Equation 5.5), one for each horizontal and vertical motion components v_i^x and v_i^y.

$$H_i = F_i +\!\!+ G_i^x +\!\!+ G_i^y. \tag{5.13}$$

The binary string H_i is a binary feature vector of dimension $D = 4L + 2M$. The BIDT descriptor captures in one single binary string intrinsic motion information of dense optical flow via tracking, as well as vertical and horizontal motion components via BWD descriptor and Prewitt operators.

5.2 Action Recognition with Bag of Features

In this section, we evaluate the accuracy of the 3DBPD, BWD and BDT descriptions. To this end, we use BoF+SVM pipeline (see Figure 5.8). From the extracted binary features (see Figure 5.8a) we generate an M-word dictionary (BoF) (see Figure 5.8b). To this end, we use k-means clustering initialised by random sampling. We use the Hamming distance to match the features with the clusters centroids. Next, we create signature histograms of the observed videos to generate the input vector of the SVM. We assign the histogram of the test video to the class which exhibits

Figure 5.9: Sample actions in the publicly available UCF50 [142] and KTH [143] datasets. The former (first two rows) contains 50 actions divided into 25 groups (splits) in a total of 6732 videos. The latter (third row) contains 6 actions of 25 people in 600 videos.

the greatest distance to the SVM hyperplane (see Figure 5.8c).

To extract the histogram, we incorporate spatio-temporal information as in [6, 7]. The technique is as follows. As Figure 5.8d) illustrates, for each video i we define a cell c over the spatio-temporal domain by dividing the video into a number of horizontal h(horizontal), v(vertical) and t(temporal) regions, denotated as $h{\times}v{\times}t$. This division gives C number of cells, for instance, $h3{\times}v1{\times}t1$ means that we divide the video into 3 horizontal, 1 vertical and 1 temporal cells, i.e., $C = 3$ cells. For each cell c, we extract a BoF and generate one signature histogram h_i^c. We normalise it using ℓ_1 norm. Next, we concatenate C signature histograms, one from each cell, into a feature vector $\mathcal{H}_i$ that represent the entire video i.

We assign the concatenated histograms $\mathcal{H}_i$ as input feature vectors for an SVM classifier trained with a χ^2-kernel. We calculate the mean χ^2 distance A_c of the set of training input feature vectors. The distance D between two videos is the distance between the corresponding histograms as follows:

$$D(h_i, h_j) = \frac{1}{2} \sum_{m=1}^{M} \frac{(h_{i,m} - h_{j,m})^2}{h_{i,m} + h_{j,m}} \tag{5.14a}$$

$$\kappa(\mathcal{H}_i, \mathcal{H}_j) = \exp\left(- \sum_{c \in C} \frac{1}{A_c} D(h_i^c, h_j^c) \right), \tag{5.14b}$$

where $h_{i,m}$ represents the entry in histogram h_i for the mth word. We associate the action in a video with a class label. We train the classifier using one-vs-all learners.

5.2.1 Experiments

The primary motivation of the proposed features is to achieve competitive performance reducing significantly computational times. Although there is a well-known trade-off between accuracy and computational time, the accuracy has to be high to be attractive otherwise the proposed binary features will have no chance to be considered as an alternative.

We use the KTH and the UCF50 datasets (see Figure 5.9) to evaluate our descriptors. The former contains 600 videos with 6 actions. Because the required testing time this dataset is particularly useful to test parameters, classifiers, complexity, etc. KTH dataset is also helpful to make inferences about the possible accuracy in more challenging datasets. The UCF dataset is a very challenging dataset with 50 actions and nearly 7000 videos. We use this dataset to demonstrate that the proposed descriptors have excellent performance under very challenging conditions.

KTH dataset

We performed several experiments using this dataset. Experiment (I), we tested our proposed descriptors 3DBPD and BWD with a different number of patterns. Experiment (II), we tested converting existing 2D patterns, such as FREAK, into 3D patterns and using the best-pair seeking to find the best pairs. Experiment(III), we compare our proposed descriptors against the state-of-the-art.

Experiment (I): For the first experiment, we modify the number of regions to compare and nature of the pairs (temporal-symmetric and spatial-symmetric) of the 3DBPD and BWD descriptors. Table 5.1 tabulates the results of this experiment.

Table 5.1: Performance on the KTH dataset of BWD and 3DBPD for different regions.

Test	CCR	Descriptor (feature vector) dimension	computational time(ms)
3DBPD 32 large regions + 8 * 4 small regions	85.64%	64	25
3DBPD 32 large regions + 8 * 8 small regions	86.12%	96	29
3DBPD 32 large regions no small regions	79.05%	**32**	**6**
BWD (spatial symmetric)	83.34%	64	7.5
BWD (time symmetric)	84.22%	64	7.5
BWD (combined 16 spatial + 16 temporal)	83.12%	**32**	**6**
BWD (combined 32 spatial + 32 temporal)	88.88%	64	7.5
BWD (combined 38 spatial + 38 temporal)	**89.35%**	76	8.3

From Table 5.1 we can see the importance of the temporal-symmetric patterns regarding accuracy for both descriptors BWD and 3DBPD descriptors. We also observe computational times improvement for the BWD descriptor.

Experiment (II): We extended existing 2D binary patterns, for instance FREAK, to the spatio-temporal domain (see Figure 5.10) to create a video descrip-

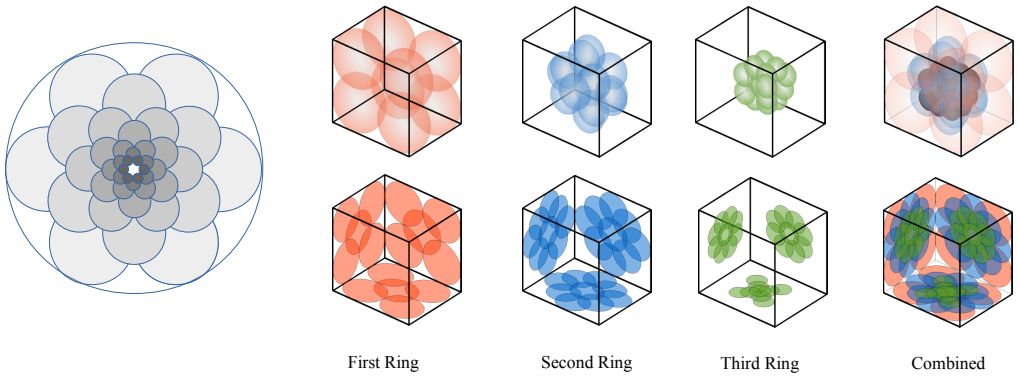


Figure 5.10: Spatio-temporal patterns to extract the extended FREAK descriptor. (left) Sample 2D FREAK patterns around a interest point. Circular regions are progressively smaller as they are closer to the interest point. (right) 3D FREAK patterns defined in the spatio-temporal domain by using ellipsoids of three different sizes organised into three rings. The first ring (outermost layer) comprises the largest ellipsoids. The second and third rings comprises progressively smaller ellipsoids emulating the 2D FREAK patterns. The top row depicts the actual spatio-temporal regions to be compared, while the bottom row depicts the projected ellipsoids on the XYT planes for visualisation purposes.

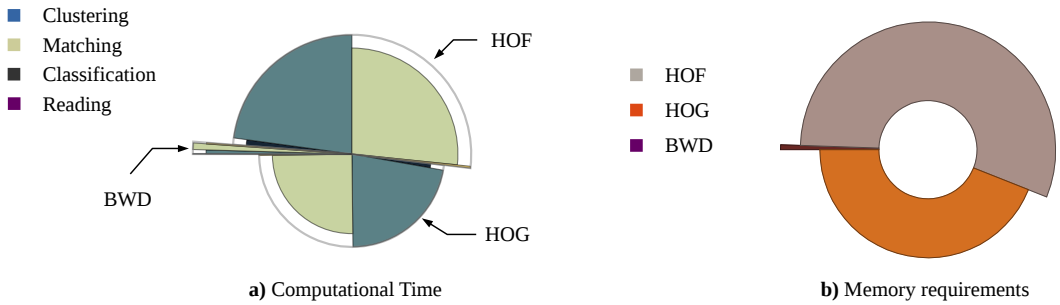
tor. As Table 5.2 tabulates, we obtained abysmal results using these extensions and employing the best-pair seeking. The extended descriptors have significantly lower performance. Results are in the range of 80% CCR with the best performance attained by the extended FREAK. These results are similar to those reported by moFREAK [144] on KTH (78-80% CCR split 9/8 and 90% CCR 24/1).

Table 5.2: Performance on the KTH dataset using STIPs with a single BoF.

Descriptors	CCR	Descriptor (feature vector) dimension
Extended FREAK	(76-81)%	32-2048
Extended BRISK	(76-80)%	32-1024
Extended ORB	(75-80)%	32-1024
Extended BRIEF(random and zig-zag scan)	(76-79)%	32-1024
3DBPD	85.65%	**64**
BWD	88.88%	**64**
BDT	**93.06%**	106

In terms of accuracy, there is a considerable difference between ORB, BRISK, BRIEF, FREAK and the descriptors we proposed that use the patterns of Figure 5.5. Therefore, we can stress that extending 2D binary descriptors to the 3D domain do not necessarily work well on action recognition.

The KTH dataset contains only six actions. And one critical aspect regarding accuracy in this dataset is that obtaining relatively good performance (e.g. circa



a) Computational Time **b)** Memory requirements

Figure 5.11: Major computational resources required by various descriptors (shown as proportions). **a)** The time of processing needed by the action recognition pipeline using BWD and single BoF. The time includes the required time to read the features, matching, clustering and classification. **b)** The memory needed to store the extracted features.

81%) is a bad start to achieve good accuracy on more challenging datasets. In order to have good accuracy on more challenging datasets, the accuracy on KTH dataset must be higher than 85%. Experiments were conducted on the UCF50 dataset (50 actions) using the extended FREAK descriptor with an accuracy of 42.76%. The proposed 3DBPD and BWD descriptors achieved on the other hand more than 50% accuracy.

Experiment (III): We compared our proposed descriptors against those of the state-of-the-art. Tables 5.3 and 5.4 summarise the accuracy while Figure 5.11 shows the computational demands. From Table 5.3, the best-reported 3DHOG descriptor has an accuracy of 91.4%. However, this descriptor is not being used in recent approaches due to its high dimensionality. This dimensionality is a critical aspect that some authors consider before selecting a descriptor. Hence, more commonly we find approaches in video analysis based on HOF descriptor for instance. On the other hand, authors discard descriptors with low accuracy no matter their fast performance, e.g. eSURF descriptor. Thus, we consider that low dimensional features with acceptable accuracy could have potential in video analysis.

Table 5.3: Performance on the KTH dataset using STIPs and the proposed descriptors with a single BoF.

Descriptor	CCR	Descriptor (feature vector) dimension	Dictionary size
3DHOG [6]	**91.4%**	960	4000
HOF [106]	89.7%	90	4000
HOG [106]	81.4%	72	4000
eSURF [105]	84.26%	384	–
BWD	88.88%	**64**	**2000**
3DBPD	85.65%	**64**	**2000**

Table 5.4: Performance on the KTH dataset using DTs and the BDT descriptor with a single BoF.

Descriptor	CCR	Descriptor (feature vector) dimensions	Dictionary size
DT [7]	89.4%	**30**	4000
HOF [7]	93.3%	108	4000
HOG [7]	87.0%	96	4000
MBH [7]	**95.0%**	192	4000
BDT	93.06%	106	**2000**

From tables 5.3 and 5.4, one can see that the proposed descriptors demonstrate competitive performance. Figure 5.11 shows that the proposed descriptors vastly outperform existing ones in terms of computational times and storage requirements.

UCF50 Dataset

We evaluate the accuracy of the proposed BWD, 3DBPD and BDT descriptors on the challenging UCF50 dataset. Tables 5.5 and 5.6 summarise the results. From the Tables, one can see that compared with HOF/HOG/PCA-gradients descriptors, the accuracy of the proposed descriptors is acceptable.

Table 5.5: Performance on the UCF50 dataset using STIPs and the BWD descriptor with a single BoF.

HOF [106]	HOG [106]	PCA-Gradients [142]	3DBPD	BWD
55.56%	52.45%	53.06%	52.25%	54.25%

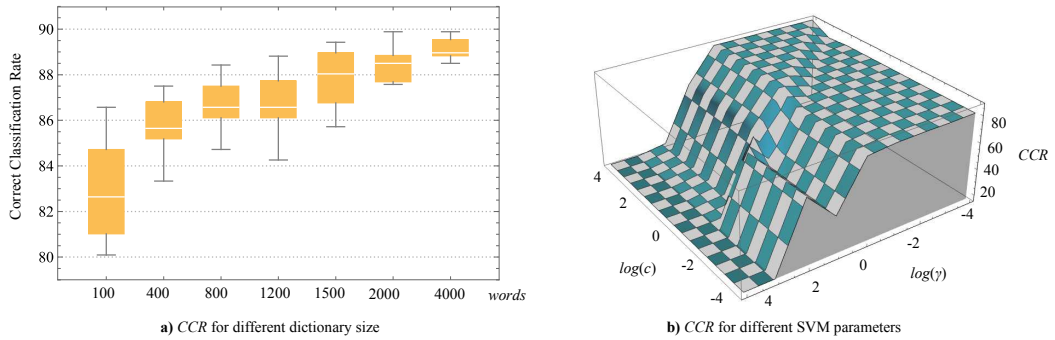
Table 5.6: Performance on the UCF50 dataset using DTs and the BDT descriptor and multi-channel BoF.

DT [7]	MBH [7]	HOG [7]	HOF [7]	BDT
67.2%	82.2%	68.0%	68.2%	67.64%

We stress the advantages of our proposed descriptors. For instance, it takes more time just to read MBH features than to generate the BoF of the BDT features. In the same line, we record the time required by the BDT+BoF+SVM pipeline (reading, clustering, matching and classifying) to be about 3 hours; just to generate the BoF of MBH features takes 18 hours. Section 5.2.3 presents discussions of Tables (5.3 – 5.6).

5.2.2 Parameters Evaluation

We use the KTH dataset as a baseline dataset to evaluate the performance of the BWD descriptor using different parameters and a single BoF. Figure 5.12a shows the CCR for different dictionary sizes. As this number increases, the CCR increases and becomes more stable. It is interesting to note that the BWD descriptor shows an excellent average performance for small dictionary sizes, e.g. 800 words. We evaluate the SVM classification using variable box constraints c and kernel scales γ. We observed no significant improvement for large c values and small γ values. Figure 5.12b shows the region with highest CCR.

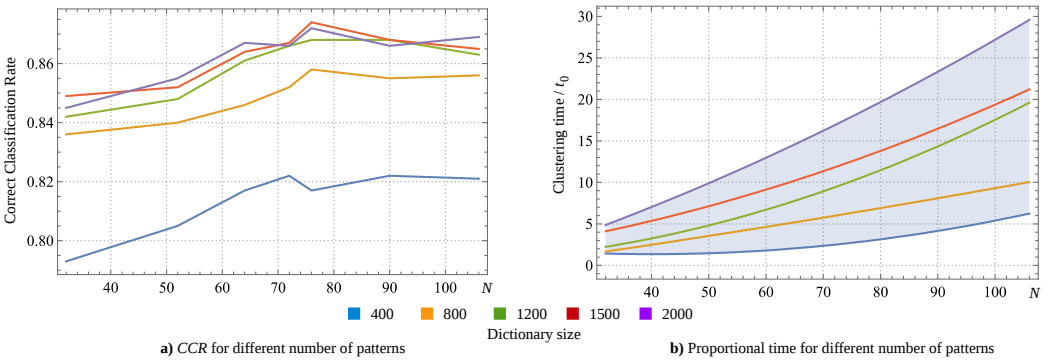


a) *CCR* for different dictionary size

b) *CCR* for different SVM parameters

Figure 5.12: CCR for different parameters. **a)** CCR for different dictionary sizes. **b)** CCR for different SVM parameters used for classification.

In a separate experiment, we evaluate the effect of selecting more or fewer patterns to create the BWD features. We randomly choose N out of 128 patterns to generate the feature. Then, we classify actions using different dictionary sizes. Figure 5.13a shows these results. We can see that using more than $N = 64$ patterns, improvements on CCR are marginal (0.5-0.8% on average) while using less than $N = 64$ patterns, the CCR values are reduced by 1-5%, on average.



a) *CCR* for different number of patterns

b) Proportional time for different number of patterns

Figure 5.13: Evaluation of different number of patterns N in the BWD descriptor. **a)** CCR for different dictionary sizes and number of patterns. **b)** Computational time required for different number of patterns and dictionary sizes.

Although we achieve slightly better performance with BWD using $N = 76$ patterns, the computational complexity is affected. Selecting more patterns has an impact on computational times. We record the computational required to generate the dictionary using a different number of patterns to observe this effect. We normalise the computational time by the runtime of the fastest clustering time, denoted by t_0. Figure 5.13b shows these computational times. Note that these times significantly increase as N increases. This increase is mainly due to the $O(n^2)$ k-means complexity and the dictionary matching time required for a larger binary feature. Therefore, $N = 64$ patterns is a good trade-off between CCR values and computational times. In general, we observe that by increasing the BWD and 3DBPD descriptor's dimensions their CCR does not increase significantly. Thus, we decide to keep both with 64 dimensions for the sake of computational complexity.

5.2.3 Discussion

Overall, the proposed descriptors demonstrate very competitive performance in terms of CCR. With respect to computational complexity, Figure 5.11 shows computational times and storage requirements compared to state-of-the-art descriptors for the BWD descriptor. Note that the times and storage requirements of BWD are just a fraction of those required by the existing state-of-the-art descriptors.

Accuracy

Based on Tables 5.3 to 5.6, the proposed descriptors demonstrate competitive results. As noticed in [6], these findings are not comparable under different setups, e.g. Wong *et al* [145], or equivalent to [60, 110, 111, 146] due to the fewer number of videos used for testing. Note that greedy methods that can compact the clusters, and which are commonly used by the other approaches compared in these tables. Those method can also help to improve the CCR of our descriptors since the action recognition pipeline is usually sensitive to the level of cluster compactness. We report the results based on an appropriate number of pair of regions ($N = 64$) and dictionary size to reduce computational times.

Clustering

We can see from Table 5.3 that the proposed BWD descriptor has the lowest dimensionality and requires the smallest dictionary size. The proposed descriptors provide distinctive features even for small dictionaries. This characteristic is an important aspect considering that clustering methods significantly increase their complexity as

a function of the feature dimensionality and dictionary's size. The proposed BWD descriptor requires a 2000-word dictionary, which is much smaller than the needed 4000-word dictionary by other approaches.

Memory requirements

The proposed BWD and BDT descriptors require **8 bytes** and **14 bytes** per feature, respectively. This compactness is an important characteristic that allows reducing memory requirements. BWD compared with the HOF descriptor, needs almost 105× less memory (see Figure 5.11b). Using BWD descriptor the 600 videos in the KTH dataset can be encoded into **4.4MB**. On average, for the UCF50 dataset, the features associated with each video require **38Kb** using the BWD descriptor, which represents 1/12 of the video size. Existing descriptors, e.g. MBH, generate features that need more than a thousand times the size of the videos in the UCF50 dataset. In our evaluations, the MBH descriptor creates more than a **Terabyte** of features for this dataset.

Processing time

The proposed descriptors significantly reduce the computational time required to classify actions (see Figure 5.11a). As noticed in [115, 116], binary features depend on two primary operations for processing: XOR and SUM. Therefore, the proposed descriptors can reduce computational times, mainly for matching and clustering operations. The MATLAB implementation of the BWD descriptor requires, on average, **7.5ms** to encode a video volume and **0.4ms** to match the features in the dictionary. These times are much shorter than those needed by the gradient-based 3DSIFT descriptor, whose MATLAB implementation requires on average 200 ms [11] to generate the feature vector.

As the amount of video data generated, e.g. on the internet, increases steadily in size every year, dedicated clusters and servers are necessary to process orientation-based features [7, 147]. Another important aspect is that real-time, on-line or large-scale applications, e.g. video retrieval, are not possible using many state-of-the-art oriented-based descriptors due to their high dimensional vectors (see Table 5.3). Concerning hardware demands, we have to consider that such descriptors might generate up to a thousand times the size of the video in features (see Figure 5.11). Eventually, this will be the amount of space necessary to scale the infrastructure if we employ these descriptors in practical scenarios.

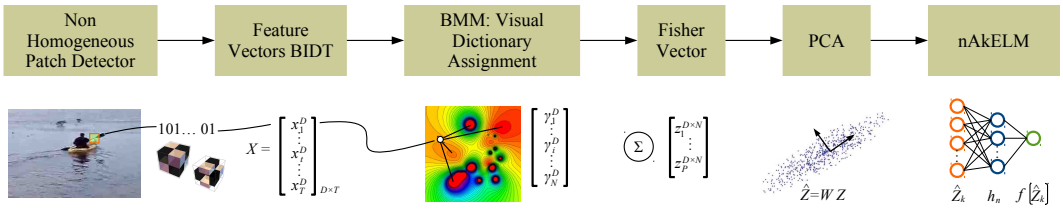


Figure 5.14: We integrate the proposed BIDT descriptor into a three-stage pipeline: In the first stage we extract binary features BIDT from the input video. The second stage consists of projecting the binary features using a probabilistic model FV-BMM. In the last stage, the projected featured dimensions are reduced using PCA and classified using a nAkELM classifier.

5.3 Action Recognition with Fisher Vectors

In Section 5.2 we employ a BoF+SVM pipeline to classify actions. However, there are better ways to perform feature-based action recognition by employing FV representations. This alternative could also be useful for BoF-based abnormal event detection (Section 4.1.4) which is BoF-based. This is because that approach requires analysing video compositions using feature representations.

Local features in image and video combined with FV representation, e.g. [141, 148, 149], has attracted much attention because of its better accuracy compared with BoF methods [150]. The FV representation has many advantages with respect to that of BoF [148, 151, 152]. It provides a more general way to define a kernel from a generative process of the data [152].

The BoF is a particular case of the FV where the gradient computation is restricted to a small number of mixture parameters of the probabilistic projection model which hinders its performance [150]. Because the FVs can represent higher order information than the BoF [149, 152], one can show that it can outperform the BoF representation in both image classification and video classification tasks [141, 149, 153, 154].

FVs aimed for binary data in image retrieval have been explored with competitive results [155]. Although there is a well-known trade-off between accuracy and computational times, projecting binary features into high dimensional vectors significantly improves performance without hindering computational efficiency.

Figure 5.14 depicts the new action recognition pipeline incorporating FVs. Instead of SVM classification, we decide to use the nAkELM classifier, that shows better performance over SVM. For this new pipeline, we employed the BIDT descriptor proposed in Section 5.1.4.

5.3.1 Fisher Vectors - Bernoulli Mixture Model

We propose to generate FVs using a BMM, similarly to the mapping of image binary features in [155]. For notation convenience, let us define x_t as the binary feature, H_i, computed by Equation 5.13. Thus $x_t \in \{0,1\}^D$ denotes a D-dimensional binary feature.

To generate the BMM, let us define an input vector X comprising a total of T binary features; $X = \{x_1, x_2, \ldots, x_T\}$. In other words, X is a binary matrix of size $D \times T$, where each row represents a binary feature and each column a particular dimension. For an N-component BMM, we define the model's distribution parameters as the set $\theta = \{w_i, \mu_{id}, i \in [1,N], d \in [1,D]\}$, where w_i is the weight of the ith BMM component, and μ_{id} is the corresponding mean across the dth dimension. The probabilistic density function for the T binary features in X is given as:

$$p(X|\theta) = \prod_{1 \leqslant t \leqslant T} p\left(x_t|\theta\right), \tag{5.15a}$$

$$p(x_t|\theta) = \sum_{1 \leqslant i \leqslant N} w_i p_i\left(x_t|\theta\right), \tag{5.15b}$$

$$p_i(x_t|\theta) = \prod_{1 \leqslant d \leqslant D} \mu_{id}^{x_{td}} \left(1-\mu_{id}\right)^{1-x_{td}}, \tag{5.15c}$$

where x_{td} represents the dth bit of x_t. The parameter set θ is estimated using the EM algorithm [39]. Specifically, the expectation step calculates the posterior probability, $\gamma_t(i) = p\left(i|x_t,\theta\right)$, of feature x_t generated by the ith BMM component as follows:

$$\gamma_t(i) = \frac{w_i p_i\left(x_t|\theta\right)}{\sum\limits_{1 \leqslant j \leqslant N} w_j p_j\left(x_t|\theta\right)}. \tag{5.16}$$

In the maximization step, the parameters are updated as follows:

$$S_i = \sum_{1 \leqslant t \leqslant T} \gamma_t(i), \;\; w_i = S_i/T, \;\; \mu_{id} = \frac{1}{S_i} \sum_{1 \leqslant t \leqslant T} \gamma_t(i) x_{td}, \tag{5.17}$$

where S_i is the zero-order statistic. Parameters w_i and μ_{id} are initialised as $1/N$ and with a uniform distribution, $U(1/4, 3/4)$, respectively.

Once the parameters converge or the EM reaches a maximum number of iterations, we proceed to map the features using the set of parameters, θ.

Deriving the FVs from the BMM follows the standard gradient derivation proposed in [148] with a GMM. A gradient vector describes the direction to which the parameters should be modified to best fit the data X. Let's describe X by the

gradient G_θ^X, also known as Fisher Score:

$$G_\theta^X = \frac{1}{T}\nabla_\theta \mathcal{L}\left(X|\theta\right), \tag{5.18a}$$

$$\mathcal{L}\left(X|\theta\right) = \log p\left(X|\theta\right). \tag{5.18b}$$

From Equation 5.15a, and assuming independence over the BMM components, the Fisher Score can be expressed in terms of the distribution parameter μ_{id} (see Appendix B.1 for derivation):

$$G_{\mu_{id}}^X = \frac{1}{T}\sum_{1\leqslant t\leqslant T}\left(\gamma_t(i)\prod_{1\leqslant d\leqslant D}\frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}\right). \tag{5.19}$$

We have to define a local metric to compare the features given by the Fisher Scores. The work in [156] proposes a specific kernel for probabilistic models based on the inner product. The idea is to derive the kernel function from a generative probabilistic model. Let us consider a parametric class of models $P(X|\theta)$, where $\theta \in \Theta$ defines the Riemannian manifold M_Θ with a local metric given by the Fisher Information matrix $F = E_X\{G^X\left(G^X\right)^\top\}$. The Fisher Score G_θ^X maps X into a new feature vector; i.e., $X \to G^X$, which is a point in the gradient space of the manifold M_Θ; specifically, the mapping is given by $G_\theta^X = \nabla_\theta \log p\left(X|\theta\right)$. The natural kernel of this mapping, κ, is the inner product between the Fisher score relative to the local Riemannian metric of two sets of features, X and Y:

$$\kappa(X,Y) = G_\theta^X F_\theta^{-1} G_\theta^Y. \tag{5.20}$$

We highlight that an inner product defines a Euclidean metric that implicitly defines a pseudo-metric in the original feature space via a second-degree polynomial expansion of the kernel [156]. Therefore, the natural kernel is a strong similarity measure in the projected space based on Euclidean distance. The information matrix required by Equation 5.20, expressed in terms of the distribution parameter μ_{id}, is calculated as follows (see Appendix B.2 for derivation):

$$F_{\mu_{id}} = \frac{Tw_i}{\mu_{id}-\mu_{id}^2}. \tag{5.21}$$

The final FV-BMM is a two-normalisation of the score concatenation $z = F_{\mu_{id}}^{1/2} G_{\mu_{id}}^X$. Firstly we use power normalisation with coefficient $\alpha \in (0,1)$, as follows:

$$f(z) = \texttt{sign}(z)|z|^\alpha. \tag{5.22}$$

Then we normalise $f(z)$ using ℓ_2 normalisation [152] to compute the final FV-BMM; i.e., $Z = \| f(z) \|_2$. For an N-component BMM of binary features with D dimensions, the FV-BMM has a dimension of $N \times D$ as a result of the concatenation of the feature projections onto each individual BMM component.

Recently [157–159] have proved that reducing the FVs' dimensions is a good practice to improve performance. Thus we apply PCA to the vectors Z to provide the transformation matrix $\hat{T}$. Thus we can split the data into training samples $\hat{Z}^X = \hat{T}Z^X$ and testing samples $\hat{Z}^Y = \hat{T}Z^Y$. For the last step, we train the nAkELM classifier with these dimensionality-reduced vectors.

5.3.2 Experiments

To evaluate the BIDT+FV-BMM pipeline, we use the UCF50 and UCF101 datasets. The next subsections present the results compared with the state-of-the-art methods in action recognition. We provide these experiment results in terms of the CCR.

UCF50 dataset

We compare BIDT+FV-BMM using the pipeline in Figure 5.14 against several state-of-the-art action recognition frameworks that have been tested on this dataset. Specifically, these frameworks are the best-performing ones that use high-order representations and double precision features [7, 141, 147], double precision features but no high-order representations [112, 142], and binary features [120]. The results of these comparisons are reported in Table 5.7, which correspond to the average CCR over the 25 splits.

Table 5.7: UCF50 dataset CCR for different methods.

Method	CCR
MIFS [147]	94.4%
IDT+MBH/HOF/HOG + FV [141]	91.2%
DT/MBH/HOF/HOG [7]	84.5%
GIST3D [112]	73.7%
c3DSIFT [142]	68.2%
MIPs [120]	72.70% †
BIDT+FV-BMM	**83.05%** †

† Binary-based approach.

From Table 5.7, we observe that BIDT+FV-BMM achieves very competitive performance compared to the best performing non-binary frameworks, outperforming GIST3D and c3DSIFT by c.a 10% and 15%, respectively. It is worth notic-

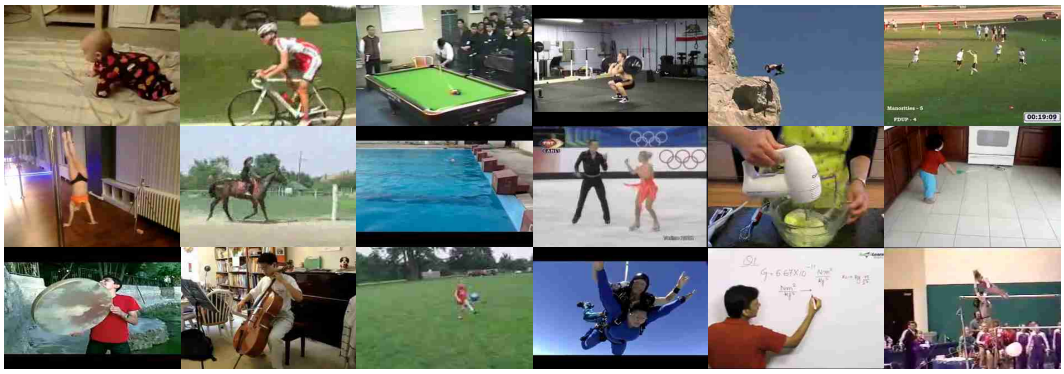


Figure 5.15: Sample actions of the publicly available UCF101 dataset. This dataset contains c.a. 13 thousand videos of 101 actions divided into 3 splits, i.e, each split has about 9k videos for training and 4k videos to test. The dataset is particularly challenging due to environmental conditions and number of classes to be tested. Most samples are captured with camera motion and complex backgrounds.

ing that BIDT+FV-BMM is about 10% more accurate than MIPs. Even though BIDT+FV-BMM is about 10% less accurate than the frameworks in [141, 147], it is important to mention that these frameworks may take days just to extract the features for this dataset. BIDT+FV-BMM requires approximately **8.5** hours to extract the features for the full UCF50 dataset and only **3.3GB** to store them, which significantly contrasts with the **846GB** required by the frameworks in [141, 147].

UCF101 dataset

This dataset is one of the most challenging ones in computer vision. It comprises 101 actions in c.a. 13 thousand videos with a large number of classes and very challenging environmental conditions, including complex backgrounds with camera motion and illumination changes. Figure 5.15 shows some example frames from this dataset. We compare BIDT+FV-BMM using the pipeline in Figure 5.14 against several state-of-the-art action recognition frameworks that have been tested on this dataset. To the best of our knowledge, no other binary framework has been tested on this dataset, therefore, our evaluations concentrate on non-binary frameworks. We evaluate 1) the CNN frameworks proposed in [8, 160, 161], 2) the framework proposed in [141], which uses high-order representations and double precision features, and 3) the framework presented in [162], which is among the best-performing ones that use double precision features but no high-order representations. The results of this evaluation, in terms of the average CCR over the 3 splits, are reported in Table 5.8.

From Table 5.8, we observe that BIDT+FV-BMM achieves competitive performance compared to the CNN frameworks. For example, BIDT+FV-BMM is only

Table 5.8: UCF101 CCR for various methods.

Method	CCR
Two Stream-CNets [8]	**88.0%**
IDT+MBH/HOF/HOG + FV [141]	85.9%
EMV-CNets [160]	79.3%
MultiRes-CNets [161]	65.4%
H3D+HOF/HOG + SVM [162]	43.9%
BIDT+FV-BMM†	71.6% †

† Binary-based approach.

7% less accurate than the CNN framework in [160]. BIDT+FV-BMM outperforms the CNN framework in [161], and the double precision feature based framework in [162], by 7% and 28%, respectively. It is important to note that, although CNN frameworks and some of those based on double precision features attain higher CCR than those attained by BIDT+FV-BMM, their computational times are considerably longer. CNN frameworks usually require High Power Computing (HPC). Even when using dedicated clusters or servers with arrays of GPUs [8], they may still take up to months for training [160]. The same drawback is shared by frameworks that employ double precision features. Even when using dedicated servers, they may take several days to process the extracted features [141, 147]. On the contrary, BIDT+FV-BMM takes only hours to process the UCF101 dataset, i.e., approximately **16 + 9** hours. Moreover, storing double precision features is highly demanding. For example, the double precision features used in [141] require around **1.7TB** of storage for the UCF101 dataset, which drastically contrasts with the **5.2GB** storage requirements of BIDT. Table 5.9 summarizes the computational demands of BIDT and FV-BMM as integrated into the pipeline of Figure 5.14 when tested on a CPU intel icore5.

Table 5.9: Total BIDT/FV-BMM demands. CPU implementation

Dataset	BIDT features storage	BIDT+FV-BMM computational time
UCF101	5.21Gb	8.78h
UCF50	3.29Gb	6.35h

From Table 5.9, we can observe that BIDT and FV-BMM require significantly low computational resources for both UCF50 and UCF101 datasets.

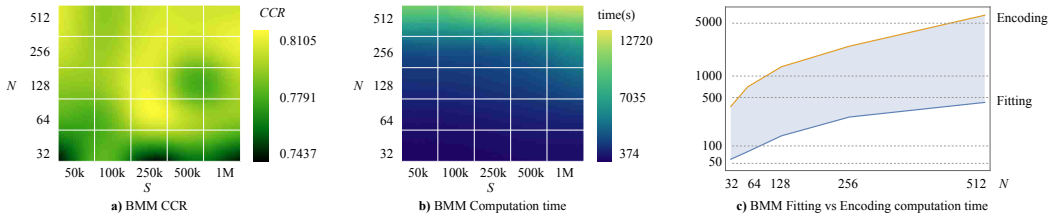


Figure 5.16: BMM parameters evaluation using UCF50 split-1. **a)** The CCR is evaluated for N-BMM components fitting X_S binary features. **b)** BMM computational time using the same setting parameters N and S. **c)** BMM Fitting vs Encoding computational time required for a $S = 100k$ BIDT features.

5.3.3 Parameters Evaluation

We use the UCF50 dataset for the evaluation of several parameters of the BMM, PCA and nAkELM. We conduct several experiments to demonstrate the benefits and characterise the proposed FV-BMM. For the first set of experiments we employ UCF50 split-1. We compute the CCR attained by the pipeline depicted in Figure 5.14 and the computational time of FV-BMM when a different number of BMM components are used and selecting a different number of BIDT features to create the model. These are the most influential parameters regarding accuracy of the proposed FV-BMM.

Experiment (I): We use N BMM components with X_S binary features vectors to be fitted, where X_S is a subset of X with S features vectors randomly selected. The results of this experiment are shown in Figure 5.16a and 5.16b. We observe that the CCR is particularly high for $N > 64$ BMM components and $S > 250k$ BIDT features vectors. As expected, computational times are also particularly high for $N > 64$ and $S > 250k$ (see Figure 5.16b). Experimentally, we find that $S = 250k$ BIDT features vectors using $N = 256$ BMM components is a good tradeoff between CCR and computational times. For these particular set of values, the computational time of FV-BMM is about 2.2 hours (see Figure 5.16b). Finally, Figure 5.16c plots the computational time of the fitting and mapping processes of FV-BMM for N BMM components fitting $S > 250k$ BIDT features vectors. From this figure, we can observe that mapping is indeed the process in FV-BMM that takes the longest and has exponential behavior with respect to the number of components.

Experiment (II): For the next batch of experiments, we fix the number of BMM components to $N = 256$ and $S = 256k$ and evaluate the CCR for M-PCA dimensional reduction, nAkELM n subsample size and C constraint factor. From Figure 5.17a, we observe that the plot becomes denser as $M > 256$, thus selecting M in that interval is a good practice to achieve good performance. From this figure,

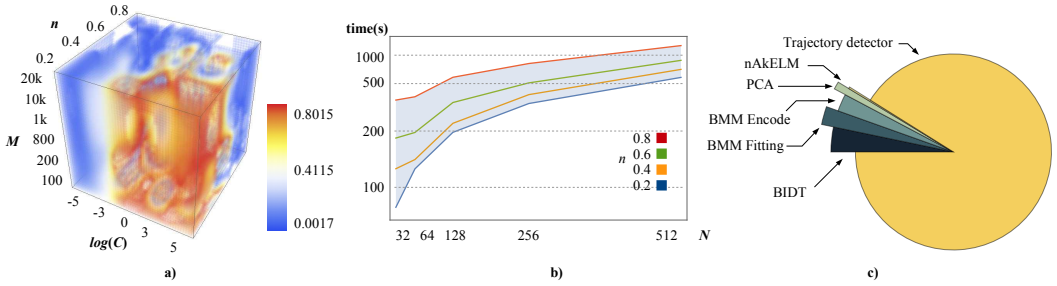


Figure 5.17: PCA/nAkELM parameter evaluation. **a)** CCR performance reducing the BMM features dimensions with M-PCA components, nAkELM subsample size n and constrain factor C. **b)** PCA/nAkELM computational time for different N-BMM components and nAkELM subsample size n. **c)** Proportional required time by the pipeline for each major individual step.

we can also observe that the CCR is higher when $n < 0.5$ and $\log(C) > 0$, thus selecting the values within that range is a good practice. From Figure 5.17b one can see that increasing/decreasing n has a minor impact on the computational time as $N < 256$, for this number of N components the nAkELM computational time is roughly the same. In this figure, we also observed that computational time increases exponentially with the N-BMM components.

For the next experiment, we record the total pipeline time when processing the UCF101 dataset for $\{N = 256, n = 0.5, M = 1000, S = 1M\}$, Figure 5.17c shows the results. The trajectory detector [141] requires most of the pipeline time, i.e. $\approx 87\%$. The proposed FV-BMM requires 8.78 hours. The nAkELM [42] requires c.a. 15 minutes, thus it can be considered negligible. From this plot, we observe that significant improvements can be made by tackling the detector's complexity.

5.3.4 Discussion

Accuracy

Tables 5.7 and 5.8 demonstrate that the proposed FV-BMM have competitive accuracy. Considering the binary nature of the proposed FV-BMM, the accuracy is notably higher and outperforms best binary-based MIPs. This improvement is worth noticing because as one can see in that table, binary-based methods trade accuracy for computational times. Let us recall that this thesis is meant to minimise that gap.

Complexity

We demonstrate that the proposed FV-BMM drastically outperform computational demands. The proposed FV-BMM requires less than **9 hours** to process the entirely UCF101 dataset using a conventional CPU machine. Therefore we **do not need** clusters, servers, arrays of GPU's, specific purpose hardware, etc. to process these video data. Regarding storage demands, the proposed BIDT requires only **22 bytes** per feature vector, which is one the benefits because the UCF50 dataset needs c.a. **3.3Gb** and UCF101 dataset **5.2Gb** of storage, respectively. This compactness demonstrates remarkable advantages regarding storage demands over existing features if we consider, for instance, that orientation-based features could require hundreds of times more storage.

As video data grows exponentially year a year, we have to explore efficient alternatives to analyse video contents. We observe that employing orientation-based features, some of the UCF101 videos required more than one **thousand times** the storage of its original size. When using the BIDT, each video requires on average **0.75** of its original size.

The binary descriptors in this chapter prove that we can achieve efficient computing for video analysis while preserving accuracy. Nevertheless, we have to highlight that minimising the gap between accuracy and computational times is challenging due to the speed-accuracy trade-off.

5.4 Conclusions

In this chapter, we presented four binary descriptors to analyze video content, the 3DBPD, the BWD, the BDT and the BIDT. These are binary descriptors that generate compact feature vectors with low complexity. They encode motion information from two sources, namely, dense optical flow tracking and temporal gradients.

We also proposed the FV-BMM, which is a binary probabilistic model that maps binary features into high dimensional spaces while keeping a generative process of the data.

Extensive evaluations for action recognition using the KTH, UCF50, and UCF101 datasets confirm the advantages of our proposed descriptors in terms of processing times and storage requirements. Results also show the higher accuracy of our binary features compared to state-of-the-art descriptors.

This page intentionally left blank.

Chapter 6

Conclusions and Future Work

This chapter discusses our contributions and essential aspects to address in the future. In this thesis, we needed to reduce computational times of fundamental aspects of video processing, such as feature extraction and high order feature representation. As a fact, abnormal event detection systems require descriptive features. However, the feature extraction mechanism may also increase computational times. To address this problem, Chapter 5 describes binary features. These features can be as accurate as double precision features for action recognition. Thus, future work could build on our binary features to detect abnormal events.

The remaining of this chapter is organised as follows: Section 6.1 presents a summary of our contributions and essential aspect to address for abnormal event detection. Section 6.2 shows a summary of our contributions and possible future trends regarding binary-based action recognition.

6.1 Abnormal Event Detection

6.1.1 Summary of our Contributions

Chapter 4 presents an online framework for video anomaly detection, i.e., ADCSF. One of the key aspects is that it processes a compact set of features based on foreground occupancy and optical flow information. To this end, the ADCSF employs a variable-sized cell structure which allows extracting features from a limited number of different support regions in a fine-to-coarse fashion. This procedure helps to process a significantly smaller number of them than those processed by dense-scanning based methods.

During the evaluations, Tables 4.3 - 4.7 reveal that there is a trade-off between computational times and detection accuracy. The ADCSF manages to detect

accuracy abnormal events while still achieving online performance. It outperforms other online methods while being very competitive among non-online ones. Therefore, we conclude that it is possible to design an online event detection system if the constituent components' complexities are kept to a minimum.

As part of the evaluations, Section 4.2.4 shows that it is possible to modify the parameters of the ADCSF considering the characteristics of the sequences. If these are known *a priori*, we can improve the accuracy. Future work aims at further enhancing the ADCSF's detection accuracy by exploiting this flexibility; specifically, by considering the optimisation of the ADCSF's parameters given particular environmental conditions. We conclude that parameter adaptation is a branch of abnormal event detection that deserves a separate study for its better comprehension and could significantly improve online methods.

Section 4.2.4 details evaluations of the ADCSF using the popular UMN, UCSD, Subway and the LV dataset. Most of the state-of-the-art video abnormal detection methods are designed and tested on datasets that poorly reflect real events commonly found in videos acquired by surveillance cameras. Therefore, we conclude that their applicability in practical situations has not been corroborated, which is essential to the design of inference mechanisms for real video abnormal detection.

Section 4.2.1 describes a new collection of surveillance videos, the LV dataset, which comprises real sequences captured by surveillance cameras under challenging environmental conditions. We conclude that future work should be tested using real video surveillance and new systems must meet online requirements to verify their effectiveness in practical scenarios.

6.1.2 Aspects to Being Improved

During the evaluations, we observed that the ADCSF does not detect some of the events, this requires attention. It identifies events when the scene's conditions are favourable. However, this is not always the case. To overcome this problem, we manually tune the parameters to evaluate our approach, but this is impractical in real scenarios.

Selecting the right parameters could become very problematic when we have thousands of cameras. Thus, ADCSF still requires elaborate formulations to be practically implemented. Automatic parameter tunning could solve this problem in order to create a practical system.

Another important aspect is that we use MATLAB for our experiments and a desktop machine. Perhaps this is not appropriate to emulate a real scenario. If we assume that multiple cameras are connected, e.g., a server, the total demands of

the ADCSF are n times the number of cameras and should not exceed the server's capacities. A better setup could be testing our system using a Raspberry Pi and see whether it still achieves online performance. In this case, the Raspberry Pi could provide more promising insights of the large-scale capabilities of the ADCSF.

6.1.3 Future Work

BoF Model

The ADCSF is BoF-based. However, we have to stress that the accuracy of sparse representations is better than the accuracy of dictionary distance metrics.

A drawback of sparse representations is the computational time. Thus, one possible improvement is to replace double-precision features and develop a binary-based sparse reconstruction model. This replacement will require adapting the feature vectors that maximise the sparse representation. Possibly, Vector of Linearly Aggregated Descriptors (VLAD) [153] could address this problem. Instead of matching the features with the BoF, we can retrieve a vector by projecting them in the dictionary similarly to the FV generation.

Experiments on the UCF50 dataset (Section 5.3.2) reveal that we can improve accuracy by more than 10% using high order feature representation. Thus a promising trend in abnormal event detection is to perform high dimensional feature projection.

Dense scanning

More features, e.g. overlapping techniques, demonstrate better performance in abnormal event detection [50]. Thus, we can perform dense or overlap scanning keeping online times by considering the memory demands and computational time of the proposed binary features.

It is well known that large dictionaries have higher detection accuracy than small ones. This variant could be explored in future work using binary features meeting online times. Figure 5.11 shows that matching double precision features requires 20× the time needed by their binary counterparts. Thus, we can easily extract 20× more features keeping the same computational time without requiring a significant amount of memory. We can take advantage of this aspect as the experiments in Section 5.2.1 revealed.

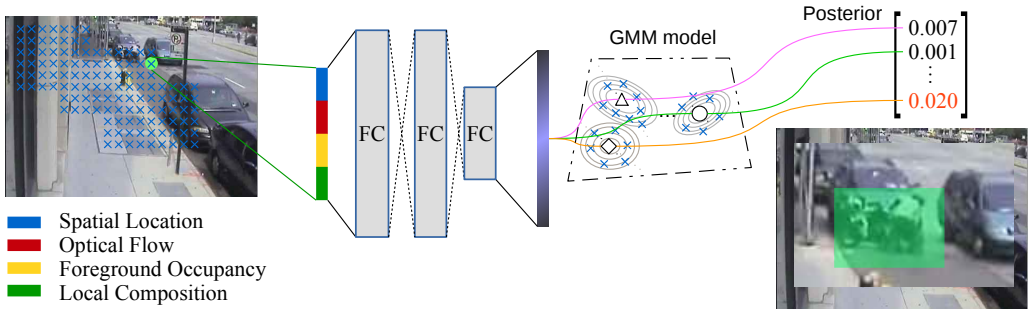


Figure 6.1: Possible end-to-end deep learning framework to detect abnormal events in video.

Deep Learning

Deep learning is fast becoming a standard for many computer vision applications. Thus, one possible manner to encode STSRs is by using an autoencoder as a global abnormal detection model. In that case, the local features should provide information of the spatial location of the STSRs.

To use a global model, the STSRs should also describe the local composition. One possible way to address this aspect is to provide as a feature vector the normalised spatial position with a zero-reference coordinate system for each encoded feature. In this case, each feature captures the centre of its composition. Therefore, each STSR captures into one single vector information of its spatio-temporal location, its local surrounding composition and its motion information (e.g. temporal gradient, optical flow and/or foreground). We can input this local feature to an autoencoder; thus we do not require separate models for each spatio-temporal location, as we can provide these features directly to a global model.

The autoencoder could be tailored with a GMM to detect abnormal events. The task is to investigate if the deep learning model can detect these events when sparse data is fed in this way. Figure 6.1 illustrates the possible method.

6.2 Action Recognition using Binary Features

6.2.1 Summary of Contributions

Chapter 5 presents four efficient video descriptors for action recognition: 3DBPD, BWD, BDT and BIDT. BWD and 3DBPD efficiently capture volumetric temporal differences in a low-dimensional binary string. The descrtiptors require two core steps, which involve compensating for the orientation and making pixel comparisons. BDT and BIDT capture motion information obtained from optical flow.

Evaluations on the UCF101 and UCF50 datasets show that BWD and BDT rank among the best state-of-art BoF-based video descriptors regarding action recognition rate while outperforming them in terms of computational times and memory requirements. Thanks to their binary nature and high descriptiveness, these descriptors can reduce more than 40× the pipeline processing time compared to the existing ones.

The binary nature of the proposed descriptors also makes them suitable for fast video processing with low hardware demands. We can perform associated computations for action recognition (e.g. clustering and matching) exceptionally efficiently even in low-cost hardware. This aspect is essential considering the vast number of applications which require feature-based video analysis. We conlude that the proposed descriptors are an attractive solution to significantly reduce processing times and memory requirements for video analysis.

Section 5.1.2 describes new patterns to generate binary features. These patterns do not require the best-pair seek formulation and outperform the existing ones, including other binary mapping techniques. The BDT uses this patterns and takes only milliseconds to be computed requiring only a memory fraction of the needed by existing state-of-the-art descriptors. We conclude that binary features can equate their double precision counterparts for video analysis tasks.

Section 5.3.1 describes FV-BMM. This binary-based probabilistic variant of the BMM maps sets of BIDTs into high dimensional FVs. This proposed model efficiently projects the features while keeping a generative process of the data. We demonstrated that the BIDT and FV-BMM can be computed remarkably fast and require a small amount of memory.

Chapter 5 narrows the gap between accuracy and computational complexity. Compared with binary-based approaches the BIDT and FV-BMM significantly enhances accuracy. We conclude that it is possible to develop real-time or online video analysis applications using binary-based features with competitive accuracy.

6.2.2 Aspects to Being Improved

Regarding accuracy, the proposed binary descriptors are consistently behind Deep Learning models that employ double precision features. The precision is around 18% lower than some of the best-reported methods. We have to address this aspect incorporating Deep Learning to classify actions.

We have used off-shelf dense trajectory detectors to extract our binary features. However, these detectors tend to be very complex. This dense trajectory detector spends up to 90% of the total computational time required by our pipelines.

By no means, the BIDT descriptor could meet online video processing if the detector requires dense optical flow. Therefore, we must either speed up the optical flow computation or replace the detector. The former naturally requires developing a fast optical flow approach, which might be very challenging. The latter might involve severely hindering our descriptor's accuracy.

6.2.3 Future Work

Binary Features - Patterns

Chapter 5 explores several patterns to compare pixel regions in order to compute the BWD and 3DBPD features. From the experiment in Section 5.2.1, we notice that specific actions (for example handclapping and handwaving of the KTH dataset) are classified better by removing some time-symmetric pairs. This observation leads us to conclude that perhaps some patterns work better for some particular actions.

One possible topic to explore is to determine what pairs are more descriptive for what sort of actions. This experiment could reveal the pairs more suitable for long or short term activities. Thus, we can design new pairs or give more priority to those that we know are associated with a particular type of movement. This more elaborate design could help to distinguish actions that look very similar, thus, reducing confusion.

Binary Features - Complexity

The BWD and 3DBPD generate features by summing pixel values in different regions. One can easily see that some of the pairs overlap others. Therefore, the implementation of the proposed descriptor is not entirely optimised because it sums values over the very same regions. One possible solution to address this problem is to use Integral Video. Integral Video is an extension of Integral Image and is an equivalent representation of summed regions that avoid their recalculation. This new variant of the descriptor requires defining the points from which we want to calculate the integral. Therefore, the descriptors may have completely different expressions but will represent equivalent features. One essential advantage of Integral Video is that no matter the STSR size, features require the same computation time.

Another straightforward improvement regarding processing times is the software implementation of our descriptors, MATLAB code could be easily replaced by a more efficient language, e.g. c.

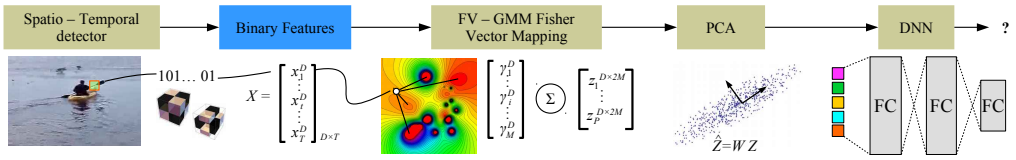


Figure 6.2: Possible end-to-end deep learning framework for action recognition using binary features.

Binary Features - Accuracy

Our descriptors only encode the temporal gradient on the grayscale video. It is well-known that video augmentation and colour encoding significantly improve accuracy. The former would require some fundamental transformations from the video, e.g. rotation, scale and interpolation. The latter would require extending the descriptor by concatenating colour information from the STSR. A binary feature could easily comprise colour information by concatenating bits that represent the most dominant colour present in the STSR. To make them robust, we have to quantise the colours and increase the number of bits used to represent them. The reason is that only three bits representing the colour components will not make a feature strong.

Deep Learning

Chapter 5 evaluates action recongition using SVM and nAkELM classifiers, as part of future work we will try to enhance accuracy via replacing the classifier by a Deep Neural Network (DNN). One paper suggests that FVs can be combined with neural networks to improve accuracy significantly [149]. We have not explored this possibility yet.

Deep learning can boost the accuracy of the nAkELM pipeline. In the most obvious way, this can be done replacing the nAkELM classifier with a DNN. The FV-BMM method requires PCA. Perhaps the DNN does not require PCA. We have to ecvaluate the accuracy without performing PCA in that case.

Finally, a major aspect is that DNNs require more data than traditional classifiers; thus we have to explore alternatives to provide the DNN with more samples. Again video augmentation could address this problem. However, we have to study which augmentations improve classification. Considering that a video provides an extra time dimension, these augmentations grow in the order of 2^{2n}, where n is augmentation factor. Therefore, significant potential contributions lie in establishing which augmentations enhance accuracy in action recognition using DNN models. Figure 6.2 illustrates the possible method.

Applications

Chapter 5 only focuses on evaluating our binary-based strategies in action recognition. However, there are a vast number of computer vision applications that could benefit from the findings of this thesis. One application that can be immediately noted is the optical flow. As Section 2.1.4 explains, the optical flow is meant to match a pixel in a local area to find the displacement vector; thus we can perform this matching using a local binary feature instead of an Summed of Squares Differences (SSD) procedure.

Another possible application is video retrieval for Big Data. Along with the video, we can save thousands of binary features with relatively low storage demands. This storage will require a fraction of the memory needed by the original video. Thus, a Big Data application could be matching small video sequences with very long video sequence whose binary features are already encoded and stored requiring memory fractions demands of the original video.

Finally, all computer vision tasks that require fast analysis of the spatio-temporal domain at any stage could benefit from the findings of this thesis.

Appendix A

Cell Structure

A.1 Construction of the Cell Structure

1. Define $y_0 > 0$ (i.e., size of the smallest square cell) and $\alpha > 1$ (i.e., growth rate of the cell size). See Fig. A.1a).

2. Adjust y_0 to $\hat{y_0}$ in order to fit an integer n number of square cells across the vertical dimension Y of the frame:

$$n = \lfloor \log_\alpha \left(Y / y_0(\alpha - 1) + 1 \right) - 1 \rceil, \tag{A.1}$$

and

$$\hat{y_0} = \Big\lfloor \frac{\alpha - 1}{\alpha^{n+1} - 1} Y \Big\rfloor, \tag{A.2}$$

3. Calculate the size of the n square cells to be created across the vertical dimension Y using the recursive equation $y_{k+1} = \alpha y_k$. For instance, for the set of parameters $\{\hat{y_0} = 10, \alpha = 1.25\}$, and a vertical dimension $Y = 160$, this recursive equation generates $n = 6$ cells of increasing sizes $\{10, 13, 20, 25, 30, 38\}$ (see Figure A.1b)).

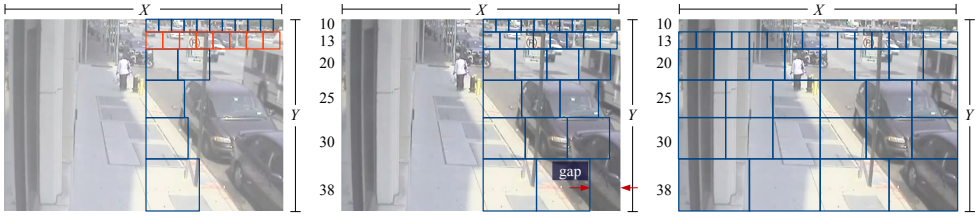
4. Starting at $X/2$, i.e., the mid point of the frame along the horizontal dimension X, populate an integer number of square cells across the X dimension, as illustrated in Figure A.1c). Repeat the same process for the remaining sizes computed in step 3) (see Figure A.1d)). In our example these sizes are $\{13, 20, 25, 30, 38\}$.

5. Fill in any horizontal gaps in order to completely cover the frame in the horizontal dimension from $X/2$ to X. This is done by adding one pixel to the horizontal dimension of the cells populated in step 4) until the cells completely

a) Define the size of the smallest square cell in the structure.

b) Populate cells in the vertical direction.

c) Populate cells in the horizontal direction, starting from $X/2$.



d) Repeat horizontal population for $\{13, 20, \ldots\}$

e) Square cells after full population.

f) Adjusted cells to completely fit the horizontal dimension.

Figure A.1: Example of cell structure generation.

cover the frame from $X/2$ to X (see Figure A.1e)). Note that due to this adjustment in the horizontal size of the cells, the final cells may not be square.

6. Cover the other half of the frame using the cell sizes computed in step 5) (see A.1f)).

7. The first row of cells comprises the smallest cells. Our experiments show that false alarms are often triggered in this first row of cells. Based on this observation, we discard the first row from the structure (see Figure A.1f)).

Appendix B

Fisher Vectors for Binary Data

B.1 Fisher Score

Let us recall the calculation of the Fisher Score, G_θ^X:

$$G_\theta^X = \frac{1}{T}\nabla_\theta \mathcal{L}\left(X|\theta\right), \tag{B.1a}$$

$$\mathcal{L}\left(X|\theta\right) = \log p\left(X|\theta\right), \tag{B.1b}$$

$$p(X|\theta) = \prod_{1\leqslant t\leqslant T} p\left(x_t|\theta\right). \tag{B.1c}$$

By substituting Eq. B.1c into Eq. B.1b, we have:

$$\mathcal{L}\left(X|\theta\right) = \log\left(\prod_{1\leqslant t\leqslant T} p\left(x_t|\theta\right)\right), \tag{B.2a}$$

$$\because \log(\prod_k a_k) = \sum_k (\log(a_k)), \tag{B.2b}$$

$$\mathcal{L}\left(X|\theta\right) = \sum_{1\leqslant t\leqslant T} \log\left(p\left(x_t|\theta\right)\right). \tag{B.2c}$$

The Fisher Score, G_θ^X, can then be expressed as:

$$G_\theta^X = \frac{1}{T}\nabla_\theta \sum_{1\leqslant t\leqslant T} \log\left(p\left(x_t|\theta\right)\right), \tag{B.3a}$$

$$= \frac{1}{T}\sum_{1\leqslant t\leqslant T} \partial_\theta \log\left(p\left(x_t|\theta\right)\right), \tag{B.3b}$$

$$= \frac{1}{T}\sum_{1\leqslant t\leqslant T} \frac{1}{p\left(x_t|\theta\right)}\partial_\theta\left(p\left(x_t|\theta\right)\right). \tag{B.3c}$$

Let us now estimate $\partial_\theta \left(p\left(x_t|\theta\right)\right)$ using $p(x_t|\theta)$ and $p_i(x_t|\theta)$:

$$p(x_t|\theta) = \sum_{1\leqslant i\leqslant N} w_i p_i\left(x_t|\theta\right), \tag{B.4a}$$

$$p_i(x_t|\theta) = \prod_{1\leqslant d\leqslant D} \mu_{id}^{x_{td}} \left(1-\mu_{id}\right)^{1-x_{td}}. \tag{B.4b}$$

Assuming independence of w_i from θ and considering parameter μ_{id} over the BMM distribution, we have:

$$\partial_\theta p(x_t|\theta) = \sum_{1\leqslant i\leqslant N} \partial_{\mu_{id}} \left(w_i p_i\left(x_t|\theta\right)\right), \tag{B.5a}$$

$$= \sum_{1\leqslant i\leqslant N} w_i \partial_{\mu_{id}} \left(p_i\left(x_t|\theta\right)\right). \tag{B.5b}$$

Using Eq. B.4b, $\partial_{\mu_{id}} \left(p_i\left(x_t|\theta\right)\right)$ is then:

$$\partial_{\mu_{id}} p_i(x_t|\theta) = \prod_{1\leqslant d\leqslant D} \partial_{\mu_{id}} \left(\mu_{id}^{x_{td}} \left(1-\mu_{id}\right)^{1-x_{td}}\right). \tag{B.6}$$

Simplifying the notation; i.e., $x_{td} \to x, \mu_{id} \to \mu$, gives us:

$$\prod_{1\leqslant d\leqslant D} \partial_\mu \left(\mu^x (1-\mu)^{1-x}\right), \tag{B.7a}$$

$$\prod_{1\leqslant d\leqslant D} x(1-\mu)^{1-x}\mu^{x-1} - (1-x)(1-\mu)^{-x}\mu^x, \tag{B.7b}$$

$$\prod_{1\leqslant d\leqslant D} (1-\mu)^{-x}(x-\mu)\mu^{x-1}, \tag{B.7c}$$

$$\prod_{1\leqslant d\leqslant D} \mu^x (1-\mu)^{1-x} \frac{x-\mu}{\mu(1-\mu)}, \tag{B.7d}$$

$$\underbrace{\prod_{1\leqslant d\leqslant D} \mu_{id}^{x_{td}} (1-\mu_{id})^{1-x_{td}}}_{p_i(x_{td}|\theta)} \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}, \tag{B.7e}$$

$$\therefore \partial_{\mu_{id}} p_i(x_t|\theta) = p_i(x_{td}|\theta) \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}. \tag{B.7f}$$

After substituting B.7f in B.5b, we have:

$$\partial_\theta p(x_t|\theta) = \sum_{1\leqslant i\leqslant N} w_i p_i(x_{td}|\theta) \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}. \tag{B.8}$$

Let us now recall the calculation of the posterior likelihood, $\gamma_t(i)$:

$$\gamma_t(i) = \frac{w_i p_i\left(x_t|\theta\right)}{\sum\limits_{1\leqslant j\leqslant N} w_j p_j\left(x_t|\theta\right)}. \tag{B.9}$$

After substituting Eq. B.9 in Eq. B.3c, we finally have:

$$\frac{1}{T}\sum_{1\leqslant t\leqslant T} \underbrace{\frac{1}{p\left(x_t|\theta\right)}\sum_{1\leqslant i\leqslant N} w_i p_i(x_{td}|\theta)}_{\gamma_t(i)} \prod_{1\leqslant d\leqslant D} \frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}, \tag{B.10a}$$

$$\therefore G^X_{\mu_{id}} = \frac{1}{T}\sum_{1\leqslant t\leqslant T}\gamma_t(i)\prod_{1\leqslant d\leqslant D}\frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}. \tag{B.10b}$$

B.2 Fisher Information Matrix

One can show that the first moment of the Fisher Score, i.e., its expected value, is 0. Thus the second moment, which corresponds to the Fisher Information, is given as follows:

$$F_{\mu_{id}} = E\left[\left(\partial_\theta \mathcal{L}\left(X|\theta\right)\right)^2|\theta\right], \tag{B.11a}$$

$$= \int_{x_t} p(x_t|\theta)\left(\partial_\theta \log p\left(X|\theta\right)\right)^2 dx_t. \tag{B.11b}$$

After simplifying the integration required by Eq. B.11a, we have:

$$p(x_t|\theta)\left(\partial_\theta \log p\left(X|\theta\right)\right)^2, \tag{B.12a}$$

$$p(x_t|\theta)\left(\sum_{1\leqslant t\leqslant T}\gamma_t(i)\prod_{1\leqslant d\leqslant D}\frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}\right)^2. \tag{B.12b}$$

Assuming that the derivative of the posterior probability of $\gamma_t(i)$ is sharply peaked, then $\gamma_t(i)^2 \approx \gamma_t(i), \forall i$. Using the posterior likelihood defined in Eq. 5.16, i.e., $\gamma_t(i) = w_i p_i(x_t|\theta)/p(x_t|\theta)$, and splitting the integral for the two possible values of x_t, we then have:

$$\int_{x_t=0}\ldots dx_t + \int_{x_t=1}\ldots dx_t. \tag{B.13}$$

The sharply-peaked derivative of the integrals in Eq. B.13 are simplified as:

$$\left.\frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}\right|_{x_t=0} = -\frac{1}{1-\mu_{id}}, \tag{B.14a}$$

$$\left.\frac{x_{td}-\mu_{id}}{\mu_{id}(1-\mu_{id})}\right|_{x_t=1} = \frac{1}{\mu_{id}}. \tag{B.14b}$$

Therefore, we have:

$$\begin{aligned}\int_{x_t=1} p(x_t|\theta) \sum_{1\leqslant t\leqslant T} \frac{\gamma_t(i)^2}{\mu_{id}^2} dx_t + \\ \int_{x_t=0} p(x_t|\theta) \sum_{1\leqslant t\leqslant T} \frac{\gamma_t(i)^2}{(1-\mu_{id})^2} dx_t,\end{aligned} \tag{B.15}$$

$$\begin{aligned}\sum_{1\leqslant t\leqslant T} \int_{x_t=1} p(x_t|\theta)\gamma_t(i) \frac{1}{\mu_{id}^2} dx_t + \\ \sum_{1\leqslant t\leqslant T} \int_{x_t=0} p(x_t|\theta)\gamma_t(i) \frac{1}{(1-\mu_{id})^2} dx_t.\end{aligned} \tag{B.16}$$

By using $p(x_t|\theta)\gamma_t(i) = w_i p_i(x_t|\theta)$, Eq. B.16 becomes:

$$\begin{aligned}\sum_{1\leqslant t\leqslant T} \int_{x_t=1} \frac{w_i p_i(x_t|\theta)}{\mu_{id}^2} dx_t + \\ \sum_{1\leqslant t\leqslant T} \int_{x_t=0} \frac{w_i p_i(x_t|\theta)}{(1-\mu_{id})^2} dx_t.\end{aligned} \tag{B.17}$$

After evaluating the integral, we have:

$$\sum_{1\leqslant t\leqslant T} \frac{w_i \mu_{id}}{\mu_{id}^2} + \sum_{1\leqslant t\leqslant T} \frac{w_i(1-\mu_{id})}{(1-\mu_{id})^2}, \tag{B.18}$$

which finally leads to:

$$T w_i \left(\frac{1}{\mu_{id}} + \frac{1}{1-\mu_{id}}\right), \tag{B.19}$$

or equivalently expressed using a single quotient:

$$F_{\mu_{id}} = \frac{T w_i}{\mu_{id} - \mu_{id}^2}. \tag{B.20}$$

Bibliography

[1] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008*, 30(3):555–560, March 2008. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.70825. vi, 2, 17, 43, 44, 64, 72, 88, 89

[2] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 65–72, Oct 2005. doi: 10.1109/VSPETS.2005.1570899. vi, 46, 47, 55, 72

[3] C. Piciarelli and G.L. Foresti. Surveillance-oriented event detection in video streams. *IEEE Intelligent Systems, 2011*, 26(3):32–41, May 2011. ISSN 1541-1672. doi: 10.1109/MIS.2010.38. 2, 3, 4, 5, 55

[4] Mehrsan Javan Roshtkhari and Martin D. Levine. An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions. *Computer Vision and Image Understanding*, 117(10):1436 – 1452, 2013. ISSN 1077-3142. doi: http://dx.doi.org/10.1016/j.cviu.2013.06.007. URL http://www.sciencedirect.com/science/article/pii/S1077314213001239. 2, 5, 38, 39, 43, 45, 55, 64, 65, 66, 70, 71, 72, 79, 83, 89, 93, 94

[5] R. Venkatesh Babu, Manu Tom, and Paras Wadekar. A survey on compressed domain video analysis techniques. *Multimedia Tools and Applications*, 75(2):1043–1078, 2016. ISSN 1573-7721. doi: 10.1007/s11042-014-2345-z. URL http://dx.doi.org/10.1007/s11042-014-2345-z. 2

[6] Alexander Klaser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC 2008-19th British Machine Vision Conference*, page 275, 2008. 2, 47, 53, 104, 115, 118, 121

[7] Heng Wang, Alexander Klaser, A.ser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1):60–79, 2013. ISSN 0920-5691. doi: 10.1007/s11263-012-0594-8. URL http://dx.doi.org/10.1007/s11263-012-0594-8. 2, 47, 48, 53, 110, 115, 119, 122, 126

[8] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf. 2, 127, 128

[9] A.A. Sodemann, M.P. Ross, and B.J. Borghetti. A review of anomaly detection in automated surveillance. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, 2012*, 42(6):1257–1272, Nov 2012. ISSN 1094-6977. doi: 10.1109/TSMCC.2012.2215319. 3, 4

[10] Oren Boiman and Michal Irani. Detecting irregularities in images and in video. *International Journal of Computer Vision*, 74(1):17–31, 2007. ISSN 0920-5691. doi: 10.1007/s11263-006-0009-9. URL http://dx.doi.org/10.1007/s11263-006-0009-9. 4, 41, 64, 66, 85, 86

[11] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM Multimedia Conference*, 2007. 6, 47, 53, 105, 122

[12] Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. Video anomaly detection based on wake motion descriptors and perspective grids. In *IEEE International Workshop on Information Forensics and Security (WIFS), 2014*, pages 209–214, Dec 2014. doi: 10.1109/WIFS.2014.7084329. 73

[13] Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. A fast binary pair-based video descriptor for action recognition. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 4185–4189, Sept 2016. doi: 10.1109/ICIP.2016.7533148.

[14] Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. Fast binary-based video descriptors for action recognition. In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, Nov 2016. doi: 10.1109/DICTA.2016.7797041.

[15] Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. The lv dataset: a realistic surveillance video dataset for abnormal event detection. In *2017 International Workshop on Biometrics and Forensics*, Mar 2017.

[16] Roberto Leyva, Victor Sanchez, and Chang-Tsun Li. Video anomaly detection with compact feature sets for online performance. *IEEE Transactions on Image Processing*, 26(7):3463–3478, July 2017. ISSN 1057-7149. doi: 10.1109/TIP.2017.2695105.

[17] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, Jul 2002. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017623. 12

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001. doi: 10.1109/CVPR.2001.990517. 13, 106

[19] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.1997.1504. URL http://www.sciencedirect.com/science/article/pii/S002200009791504X. 13

[20] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. ISBN 1848829345, 9781848829343. 13, 14

[21] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision (ICCV), 2005*, volume 2, pages 1508–1515 Vol. 2, Oct 2005. doi: 10.1109/ICCV.2005.104. 14, 70, 76

[22] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1623264.1623280. 15, 17

[23] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996. ISBN 0-8018-5414-8. 16

[24] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, Mar 2011. ISSN 1573-1405. doi: 10.1007/s11263-010-0390-2. URL https://doi.org/10.1007/s11263-010-0390-2. 17

[25] Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers, and Horst Bischof. Anisotropic huber-l1 optical flow. In *Proceedings of the British Machine Vision Conference (BMVC)*, London, UK, September 2009. URL http://gpu4vision.icg.tugraz.at/papers/2009/werlberger_bmvc2009.pdf. to appear. 17

[26] Andres Bruhn, Joachim Weickert, and Christoph Schnorr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, Feb 2005. ISSN 1573-1405. doi: 10.1023/B:VISI.0000045324.43199.43. URL https://doi.org/10.1023/B:VISI.0000045324.43199.43. 17

[27] Werner Trobin, Thomas Pock, Daniel Cremers, and Horst Bischof. *An Unbiased Second-Order Prior for High-Accuracy Motion Estimation*, pages 396–405. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-69321-5. doi: 10.1007/978-3-540-69321-5_40. URL https://doi.org/10.1007/978-3-540-69321-5_40. 17

[28] A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1663–1668, Sept 2009. doi: 10.1109/ICCV.2009.5459375. 17

[29] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185 – 203, 1981. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(81)90024-2. URL http://www.sciencedirect.com/science/article/pii/0004370281900242. 17

[30] G. Le Besnerais and F. Champagnat. Dense optical flow by iterative local window registration. In *IEEE International Conference on Image Processing 2005*, volume 1, pages I–137–40, Sept 2005. doi: 10.1109/ICIP.2005.1529706. 17

[31] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In AleÅą Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision âĂŞ ECCV 2006*, volume 3952 of *Lecture Notes in Computer Science*, pages 428–441. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-33834-5. doi: 10.1007/11744047_33. URL http://dx.doi.org/10.1007/11744047_33. 18, 19, 77, 113

[32] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178, 2006. doi: 10.1109/CVPR.2006.68. 20, 79

[33] Gabriela Csurka and Florent Perronnin. *Fisher Vectors: Beyond Bag-of-Visual-Words Image Representations*, pages 28–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-25382-9. doi: 10.1007/978-3-642-25382-9_2. URL https://doi.org/10.1007/978-3-642-25382-9_2. 21

[34] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759. 22

[35] Alexander Ly, Maarten Marsman, Josine Verhagen, Raoul P.P.P. Grasman, and Eric-Jan Wagenmakers. A tutorial on fisher information. *Journal of Mathematical Psychology*, 80(Supplement C):40 – 55, 2017. ISSN 0022-2496. doi: https://doi.org/10.1016/j.jmp.2017.05.006. URL http://www.sciencedirect.com/science/article/pii/S0022249617301396. 23

[36] Douglas Reynolds. *Gaussian Mixture Models*, pages 659–663. Springer US, Boston, MA, 2009. ISBN 978-0-387-73003-5. doi: 10.1007/978-0-387-73003-5_196. URL https://doi.org/10.1007/978-0-387-73003-5_196. 23

[37] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977. 25

[38] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029. 26, 27, 28, 29, 33, 34

[39] A. Juan and E. Vidal. Bernoulli mixture models for binary images. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 367–370 Vol.3, Aug 2004. doi: 10.1109/ICPR.2004.1334543. 27, 124

[40] Bertrand Clarke, Ernest Fokoue, and Hao Helen Zhang. *Principles and Theory for Data Mining and Machine Learning*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 0387981349, 9780387981345. 28, 30

[41] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8. 30

[42] Alexandros Iosifidis and Moncef Gabbouj. On the kernel extreme learning machine speedup. *Pattern Recognition Letters*, 68, Part 1:205 – 210, 2015. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2015.09.015. URL http://www.sciencedirect.com/science/article/pii/S0167865515003256. 34, 130

[43] Yanhao Zhang, Lei Qin, Hongxun Yao, and Qingming Huang. Abnormal crowd behavior detection based on social attribute-aware force model. In *IEEE International Conference on Image Processing (ICIP), 2012*, pages 2689–2692, Sept 2012. doi: 10.1109/ICIP.2012.6467453. 38

[44] Andrei Zaharescu and Richard Wildes. Anomalous behaviour detection using spatiotemporal oriented energies, subset inclusion histogram comparison and event-driven processing. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision âĂŞ ECCV 2010*, volume 6311 of *Lecture Notes in Computer Science*, pages 563–576. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15548-2. doi: 10.1007/978-3-642-15549-9_41. URL http://dx.doi.org/10.1007/978-3-642-15549-9_41. 42, 85, 86, 89

[45] M. Thida, H. L. Eng, and P. Remagnino. Laplacian eigenmap with temporal constraints for local abnormality detection in crowded scenes. *IEEE Transactions on Cybernetics, 2013*, 43(6):2147–2156, Dec 2013. ISSN 2168-2267. doi: 10.1109/TCYB.2013.2242059. 38, 42, 55, 56, 61, 64, 66

[46] S. Biswas and R.V. Babu. Real time anomaly detection in h.264 compressed videos. In *National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2013*, pages 1–4, Dec 2013. doi: 10.1109/NCVPRIPG.2013.6776164. 38, 45, 46, 89, 90, 92, 93, 94, 95, 96, 97, 98

[47] Cewu Lu, Jianping Shi, and Jiaya Jia. Abnormal event detection at 150 fps in matlab. In *IEEE International Conference on Computer Vision (ICCV), 2013*, pages 2720–2727, Dec 2013. doi: 10.1109/ICCV.2013.338. 38, 44, 45, 46, 71, 72, 89, 90, 92, 93, 94, 95, 96, 97, 98

[48] D. Ryan, S. Denman, C. Fookes, and S. Sridharan. Textures of optical flow for real-time anomaly detection in crowds. In *IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS), 2011*, pages 230–235, Aug 2011. doi: 10.1109/AVSS.2011.6027327. 38

[49] B. Antic and B. Ommer. Video parsing for abnormality detection. In *IEEE International Conference on Computer Vision (ICCV), 2011*, pages 2415–2422, Nov 2011. doi: 10.1109/ICCV.2011.6126525. 38, 92

[50] Marco Bertini, Alberto Del Bimbo, and Lorenzo Seidenari. Multi-scale and real-time non-parametric approach for anomaly detection and localization. *Computer Vision and Image Understanding*, 116(3):320 – 329, 2012. ISSN 1077-3142. doi: http://dx.doi.org/10.1016/j.cviu.2011.09.009. URL http://www.sciencedirect.com/science/article/pii/S1077314211002104. Special issue on Semantic Understanding of Human Behaviors in Image Sequences. 38, 39, 43, 45, 46, 58, 70, 89, 135

[51] M. Bertini, A. Del Bimbo, and L. Seidenari. Scene and crowd behaviour analysis with local space-time descriptors. In *International Symposium on Communications Control and Signal Processing (ISCCSP), 2012*, pages 1–6, May 2012. doi: 10.1109/ISCCSP.2012.6217857. 38, 43, 70, 83

[52] H. Dee and D. Hogg. Detecting inexplicable behaviour. In *Proceedings of the British Machine Vision Conference*, pages 50.1–50.10. BMVA Press, 2004. ISBN 1-901725-25-1. doi:10.5244/C.18.50. 39

[53] D. Makris and T. Ellis. Learning semantic scene models from observing activity in visual surveillance. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, 2005*, 35(3):397–408, June 2005. ISSN 1083-4419. doi: 10.1109/TSMCB.2005.846652. 40

[54] F. Porikli and T. Haga. Event detection by eigenvector decomposition using object and frame features. In *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW), 2004*, pages 114–114, June 2004. doi: 10.1109/CVPR.2004.61. 40

[55] C. Piciarelli and G.L. Foresti. On-line trajectory clustering for anomalous events detection. *Pattern Recognition Letters*, 27(15):1835 – 1842, 2006. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2006.02.004. URL http://www.sciencedirect.com/science/article/pii/S0167865506000432. Vision for Crime Detection and Prevention. 40

[56] C. Piciarelli, C. Micheloni, and G.L. Foresti. Trajectory-based anomalous event detection. *IEEE Transactions on Circuits and Systems for Video Technology, 2008*, 18 (11):1544–1554, Nov 2008. ISSN 1051-8215. doi: 10.1109/TCSVT.2008.2005599. 40

[57] H. Li, A. Achim, and D. Bull. Unsupervised video anomaly detection using feature clustering. *Signal Processing*, 6(5):521–533, July 2012. ISSN 1751-9675. doi: 10.1049/iet-spr.2011.0074. 40

[58] I. Ivanov, F. Dufaux, T. M. Ha, and T. Ebrahimi. Towards generic detection of unusual events in video surveillance. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2009*, pages 61–66, Sept 2009. doi: 10.1109/AVSS.2009.63. 40

[59] Teng Zhang, A. Wiliem, and B.C. Lovell. Region-based anomaly localisation in crowded scenes via trajectory analysis and path prediction. In *International Conference on Digital Image Computing: Techniques and Applications (DICTA), 2013*, pages 1–7, Nov 2013. doi: 10.1109/DICTA.2013.6691519. 40

[60] Xinghao Jiang, Tanfeng Sun, Bing Feng, and Chengming Jiang. A space-time surf descriptor and its application to action recognition with video words. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, volume 3, pages 1911–1915, July 2011. doi: 10.1109/FSKD.2011.6019848. 40, 121

[61] Simone Calderara, Uri Heinemann, Andrea Prati, Rita Cucchiara, and Naftali Tishby. Detecting anomalies in peopleâĂŹs trajectories using spectral graph analysis. *Computer Vision and Image Understanding*, 115(8):1099 – 1111, 2011. ISSN 1077-3142. doi: http://dx.doi.org/10.1016/j.cviu.2011.03.003. URL http://www.sciencedirect.com/science/article/pii/S1077314211000919. 40

[62] Frederick Tung, John S. Zelek, and David A. Clausi. Goal-based trajectory analysis for unusual behaviour detection in intelligent surveillance. *Image and Vision Computing*, 29(4):230 – 240, 2011. ISSN 0262-8856. doi: http://dx.doi.org/10.1016/j.imavis.2010.11.003. URL http://www.sciencedirect.com/science/article/pii/S026288561000154X. 40

[63] X. Mo, V. Monga, R. Bala, and Z. Fan. A joint sparsity model for video anomaly detection. In *Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), 2012*, pages 1969–1973, Nov 2012. doi: 10.1109/ACSSC.2012.6489384. 40

[64] X. Mo, V. Monga, R. Bala, and Z. Fan. Adaptive sparse representations for video anomaly detection. *IEEE Transactions on Circuits and Systems for Video Technology, 2014*, 24(4):631–645, April 2014. ISSN 1051-8215. doi: 10.1109/TCSVT.2013.2280061. 40

[65] Hua Zhong, Jianbo Shi, and M. Visontai. Detecting unusual activity in video. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2004*, volume 2, pages II–819–II–826 Vol.2, June 2004. doi: 10.1109/CVPR.2004.1315249. 40

[66] C. E. Au, S. Skaff, and J. J. Clark. Anomaly detection for video surveillance applications. In *International Conference on Pattern Recognition (ICPR), 2006*, volume 4, pages 888–891, 2006. doi: 10.1109/ICPR.2006.273. 41

[67] E. L. Andrade, S. Blunsden, and R. B. Fisher. Hidden markov models for optical flow analysis in crowds. In *International Conference on Pattern Recognition (ICPR), 2006*, volume 1, pages 460–463, 2006. doi: 10.1109/ICPR.2006.621. 41

[68] F. Archetti, C. E. Manfredotti, M. Matteucci, V. Messina, and D. G. Sorrenti. Parallel first-order markov chain for on-line anomaly detection in traffic video surveillance. In *The Institution of Engineering and Technology Conference on Crime and Security, 2006*, pages 582–587, June 2006. 41

[69] I. Pruteanu-Malinici and L. Carin. Infinite hidden markov models for unusual-event detection in video. *IEEE Transactions on Image Processing, 2008*, 17(5):811–822, May 2008. ISSN 1057-7149. doi: 10.1109/TIP.2008.919359. 41

[70] S. Ali and M. Shah. A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007*, pages 1–6, June 2007. doi: 10.1109/CVPR.2007.382977. 41, 85, 88

[71] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009*, pages 935–942, June 2009. doi: 10.1109/CVPR.2009.5206641. 41

[72] L. Kratz and K. Nishino. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009*, pages 1446–1453, June 2009. doi: 10.1109/CVPR.2009.5206771. 41, 55, 79

[73] Huan Wang, Ruiqing Fu, Nannan Li, Guoyuan Liang, and Xinyu Wu. Anomaly detection in crowds assisted by scene perspective projection correction. In *IEEE International Conference on Information Science and Technology (ICIST), 2014*, pages 686–689, April 2014. doi: 10.1109/ICIST.2014.6920570. 41, 48

[74] Y. Benezeth, P.-M. Jodoin, V. Saligrama, and C. Rosenberger. Abnormal events detection based on spatio-temporal co-occurences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009*, pages 2458–2465, June 2009. doi: 10.1109/CVPR.2009.5206686. 41, 79

[75] Chen Change Loy, Tao Xiang, and Shaogang Gong. Modelling multi-object activity by gaussian processes. In *BMVC*, pages 1–11, 2009. 41, 85

[76] Jaechul Kim and K. Grauman. Observe locally, infer globally: A space-time mrf for detecting abnormal activities with incremental updates. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009*, pages 2921–2928, June 2009. doi: 10.1109/CVPR.2009.5206569. 42, 55, 56

[77] V. Mahadevan, Weixin Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010*, pages 1975–1981, June 2010. doi: 10.1109/CVPR.2010.5539872. 42, 55, 64, 66, 85, 88, 89, 92

[78] Weixin Li, V. Mahadevan, and N. Vasconcelos. Anomaly detection and localization in crowded scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014*, 36(1):18–32, Jan 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.111. 42, 79, 89, 92, 94

[79] J. Feng, C. Zhang, and P. Hao. Online learning with self-organizing maps for anomaly detection in crowd scenes. In *International Conference on Pattern Recognition (ICPR), 2010*, pages 3599–3602, Aug 2010. doi: 10.1109/ICPR.2010.878. 42

[80] Tong Yubing, Faouzi Alaya Cheikh, Fahad Fazal Elahi Guraya, Hubert Konik, and Alain Trémeau. A spatiotemporal saliency model for video surveillance. *Cognitive Computation*, 3(1):241–263, 2011. ISSN 1866-9964. doi: 10.1007/s12559-010-9094-8. URL http://dx.doi.org/10.1007/s12559-010-9094-8. 42

[81] V. Reddy, C. Sanderson, and B.C. Lovell. Improved anomaly detection in crowded scenes via cell-based analysis of foreground speed, size and texture. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2011*, pages 55–61, June 2011. doi: 10.1109/CVPRW.2011.5981799. 42, 45, 46, 58, 70, 71, 72, 73, 75, 79, 82, 89

[82] Chen Change Loy, Tao Xiang, and Shaogang Gong. Detecting and discriminating behavioural anomalies. *Pattern Recognition*, 44(1):117 – 132, 2011. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2010.07.023. URL http://www.sciencedirect.com/science/article/pii/S0031320310003626. 42

[83] Huiwen Guo, Xinyu Wu, Nannan Li, Ruiqing Fu, Guoyuan Liang, and Wei Feng. Anomaly detection and localization in crowded scenes using short-term trajectories. In

IEEE International Conference on Robotics and Biomimetics (ROBIO), 2013, pages 245–249, Dec 2013. doi: 10.1109/ROBIO.2013.6739466. 42, 55, 73

[84] Bin Zhao, Li Fei-Fei, and E.P. Xing. Online detection of unusual events in videos via dynamic sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011*, pages 3313–3320, June 2011. doi: 10.1109/CVPR.2011.5995524. 42, 95

[85] Y. Cong, J. Yuan, and Y. Tang. Video anomaly search in crowded scenes via spatio-temporal motion context. *IEEE Transactions on Information Forensics and Security, 2013*, 8(10):1590–1599, Oct 2013. ISSN 1556-6013. doi: 10.1109/TIFS.2013.2272243. 42, 45, 46, 89

[86] Yang Cong, Junsong Yuan, and Ji Liu. Abnormal event detection in crowded scenes using sparse representation. *Pattern Recognition*, 46(7):1851 – 1864, 2013. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2012.11.021. URL http://www.sciencedirect.com/science/article/pii/S0031320312005055. 42

[87] Yang Cong, Junsong Yuan, and Ji Liu. Sparse reconstruction cost for abnormal event detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011*, pages 3449–3456, June 2011. doi: 10.1109/CVPR.2011.5995434. 42, 71, 92, 93

[88] V. Saligrama and Zhu Chen. Video anomaly detection based on local statistical aggregates. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012*, pages 2112–2119, June 2012. doi: 10.1109/CVPR.2012.6247917. 42, 45, 71

[89] L. Tran, C. Navasca, and J. Luo. Video detection anomaly via low-rank and sparse decompositions. In *Western New York Image Processing Workshop (WNYIPW), 2012*, pages 17–20, Nov 2012. doi: 10.1109/WNYIPW.2012.6466649. 43

[90] Xiaobin Zhu, Jing Liu, Jinqiao Wang, Changsheng Li, and Hanqing Lu. Sparse representation for robust abnormality detection in crowded scenes. *Pattern Recognition*, 47(5):1791 – 1799, 2014. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2013.11.018. URL http://www.sciencedirect.com/science/article/pii/S0031320313005049. 43, 45, 89, 92, 93

[91] Z. Zhu, J. Wang, and N. Yu. Anomaly detection via 3d-hof and fast double sparse representation. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 286–290, Sept 2016. doi: 10.1109/ICIP.2016.7532364. 43, 45

[92] H. Mousavi, S. Mohammadi, A. Perina, R. Chellali, and V. Murino. Analyzing tracklets for the detection of abnormal crowd behavior. In *IEEE Winter Conference on Applications of Computer Vision (WACV), 2015*, pages 148–155, Jan 2015. doi: 10.1109/WACV.2015.27. 43, 45, 46

[93] M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor. Holistic features for real-time crowd behaviour anomaly detection. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 918–922, Sept 2016. doi: 10.1109/ICIP.2016.7532491. 43

[94] Dan Xu, Xinyu Wu, Dezhen Song, Nannan Li, and Yen-Lun Chen. Hierarchical activity discovery within spatio-temporal context for video anomaly detection. In *IEEE International Conference on Image Processing (ICIP), 2013*, pages 3597–3601, Sept 2013. doi: 10.1109/ICIP.2013.6738742. 43

[95] J. Feng, C. Zhang, and P. Hao. Online anomaly detection in videos by clustering dynamic exemplars. In *IEEE International Conference on Image Processing (ICIP), 2012*, pages 3097–3100, Sept 2012. doi: 10.1109/ICIP.2012.6467555. 43

[96] N. Li, H. Guo, D. Xu, and X. Wu. Multi-scale analysis of contextual information within spatio-temporal video volumes for anomaly detection. In *IEEE International Conference on Image Processing (ICIP), 2014*, pages 2363–2367, Oct 2014. doi: 10.1109/ICIP.2014.7025479. 43

[97] M.J. Roshtkhari and M.D. Levine. Online dominant and anomalous behavior detection in videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013*, pages 2611–2618, June 2013. doi: 10.1109/CVPR.2013.337. 43, 45, 55, 70, 72

[98] K. Cheng, Y. Chen, and W. Fang. Gaussian process regression-based video anomaly detection and localization with hierarchical feature representation. *IEEE Transactions on Image Processing, 2015*, 24(12):5288–5301, Dec 2015. ISSN 1057-7149. doi: 10.1109/TIP.2015.2479561. 43, 45, 70, 72, 76, 79, 89, 93, 95

[99] Kai-Wen Cheng, Yie-Tarng Chen, and Wen-Hsien Fang. Video anomaly detection and localization using hierarchical feature representation and gaussian process regression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015*, pages 2909–2917, June 2015. doi: 10.1109/CVPR.2015.7298909. 43, 45, 70, 89

[100] Y. Zhao, L. Zhou, K. Fu, and J. Yang. Abnormal event detection using spatio-temporal feature and nonnegative locality-constrained linear coding. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3354–3358, Sept 2016. doi: 10.1109/ICIP.2016.7532981. 43

[101] S. Biswas and R. Venkatesh Babu. Sparse representation based anomaly detection using homv in h.264 compressed videos. In *International Conference on Signal Processing and Communications (SPCOM), 2014*, pages 1–6, July 2014. doi: 10.1109/SPCOM.2014.6984003. 45

[102] Yang Hu, Yangmuzi Zhang, and L.S. Davis. Unsupervised abnormal crowd activity detection using semiparametric scan statistic. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2013*, pages 767–774, June 2013. doi: 10.1109/CVPRW.2013.115. 45, 89, 92, 93, 94, 95

[103] I. Laptev and T. Lindeberg. Space-time interest points. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 432–439 vol.1, Oct 2003. doi: 10.1109/ICCV.2003.1238378. 46, 76

[104] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 47

[105] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision âĂŞ ECCV 2008*, volume 5303 of *Lecture Notes in Computer Science*, pages 650–663. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88685-3. doi: 10.1007/978-3-540-88688-4_48. URL http://dx.doi.org/10.1007/978-3-540-88688-4_48. 47, 118

[106] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008. doi: 10.1109/CVPR.2008.4587756. 47, 72, 104, 118, 119

[107] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC 2009-British Machine Vision Conference*, pages 124–1. BMVA Press, 2009. 48

[108] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4305–4314, June 2015. doi: 10.1109/CVPR.2015.7299059. 48

[109] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001. ISSN 1573-1405. doi: 10.1023/A:1011139631724. URL http://dx.doi.org/10.1023/A:1011139631724. 48

[110] Ling Shao, Ruoyun Gao, Yan Liu, and Hui Zhang. Transform based spatio-temporal descriptors for human action recognition. *Neurocomputing*, 74(6):962 – 973, 2011. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2010.11.013. URL http://www.sciencedirect.com/science/article/pii/S0925231210004753. 48, 121

[111] Ling Shao and Ruoyun Gao. A wavelet based local descriptor for human action recognition. In *BMVC*, pages 1–10, 2010. 48, 121

[112] Berkan Solmaz, Shayan Modiri Assari, and Mubarak Shah. Classifying web videos using a global video descriptor. *Machine Vision and Applications*, 24(7):1473–1485, 2013. ISSN 1432-1769. doi: 10.1007/s00138-012-0449-x. URL http://dx.doi.org/10.1007/s00138-012-0449-x. 48, 126

[113] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision âĂŞ ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 778–792. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15560-4. doi: 10.1007/978-3-642-15561-1_56. URL http://dx.doi.org/10.1007/978-3-642-15561-1_56. 48, 53

[114] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571, Nov 2011. doi: 10.1109/ICCV.2011.6126544. 48, 52, 106

[115] S. Leutenegger, M. Chli, and R.Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555, Nov 2011. doi: 10.1109/ICCV.2011.6126542. 49, 103, 104, 122

[116] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517, June 2012. doi: 10.1109/CVPR.2012.6247715. 48, 49, 52, 53, 103, 106, 122

[117] Paul L. Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291 – 307, 1999. ISSN 1077-3142. doi: http://dx.doi.org/10.1006/cviu.1998.0719. URL http://www.sciencedirect.com/science/article/pii/S1077314298907196. 48, 104

[118] Yunqian Ma and P. Cisar. Event detection using local binary pattern based dynamic textures. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 38–44, June 2009. doi: 10.1109/CVPRW.2009.5204204. 50

[119] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1509–1517. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3749-locality-sensitive-binary-codes-from-shift-invariant-kernels.pdf. 51

[120] Orit Kliper-Gross, Yaron Gurovich, Tal Hassner, and Lior Wolf. *Motion Interchange Patterns for Action Recognition in Unconstrained Videos*, pages 256–269. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-33783-3. doi: 10.1007/978-3-642-33783-3_19. URL http://dx.doi.org/10.1007/978-3-642-33783-3_19. 51, 126

[121] L. Yeffet and L. Wolf. Local trinary patterns for human action recognition. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 492–497, Sept 2009. doi: 10.1109/ICCV.2009.5459201. 51

[122] Engin Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(5):815–830, May 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2009.77. 53

[123] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74(18):3823 – 3831, 2011. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2011.07.024. URL http://www.sciencedirect.com/science/article/pii/S0925231211004668. 55, 73

[124] Hajer Fradi and Jean-Luc Dugelay. Towards crowd density-aware video surveillance applications. *Information Fusion*, 24(0):3 – 15, 2015. ISSN 1566-2535. doi: http://dx.doi.org/10.1016/j.inffus.2014.09.005. URL http://www.sciencedirect.com/science/article/pii/S1566253514001055. 55, 73

[125] Alceu Ferraz Costa, Gabriel Humpire-Mamani, and Agma Juci Machado Traina. An efficient algorithm for fractal analysis of textures. In *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*, pages 39–46. IEEE, 2012. 60

[126] Ucsd anomaly detection dataset. http://www.svcl.ucsd.edu/projects/anomaly/dataset.htm. 63, 92

[127] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. ISBN 0070131511. 67

[128] O.P. Popoola and Kejun Wang. Video-based abnormal human behavior recognition;a review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,2012*, 42(6):865–878, Nov 2012. ISSN 1094-6977. doi: 10.1109/TSMCC.2011.2178594. 72

[129] A. B. Chan, Zhang-Sheng John Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008*, pages 1–7, June 2008. doi: 10.1109/CVPR.2008.4587569. 73

[130] Y. Wang, P. M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar. Cdnet 2014: An expanded change detection benchmark dataset. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 393–400, June 2014. doi: 10.1109/CVPRW.2014.126. 75

[131] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin. Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing*, 24(1):359–373, Jan 2015. ISSN 1057-7149. doi: 10.1109/TIP.2014.2378053.

[132] S. E. Ebadi, V. G. Ones, and E. Izquierdo. Efficient background subtraction with low-rank and sparse matrix decomposition. In *2015 IEEE International Conference*

on Image Processing (ICIP), pages 4863–4867, Sept 2015. doi: 10.1109/ICIP.2015.7351731.

[133] H. Ramadan and H. Tairi. Automatic human segmentation in video using convex active contours. In *2016 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*, pages 184–189, March 2016. doi: 10.1109/CGiV.2016.43. 75

[134] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer, 2002. 75

[135] Univeristy of minnesota dataset. http://mha.cs.umn.edu/proj_events.shtml. 85, 88

[136] Opencv matlab code generator. https://github.com/Itseez/opencv_contrib/tree/master/modules/matlab. 90

[137] http://shijianping.me/. 90

[138] Kernel density estimator. http://uk.mathworks.com/matlabcentral/fileexchange/14034-kernel-density-estimator. 90

[139] Video parsing for abnormality detection. http://hciweb.iwr.uni-heidelberg.de/compvis/research/parsing. 92

[140] S. Saha and V. Demoulin. Aloha: An efficient binary descriptor based on haar features. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 2345–2348, Sept 2012. doi: 10.1109/ICIP.2012.6467367. 106, 107

[141] H. Wang and C. Schmid. Action recognition with improved trajectories. In *2013 IEEE International Conference on Computer Vision*, pages 3551–3558, Dec 2013. doi: 10.1109/ICCV.2013.441. 112, 123, 126, 127, 128, 130

[142] Kishore K. Reddy and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013. ISSN 1432-1769. doi: 10.1007/s00138-012-0450-4. URL http://dx.doi.org/10.1007/s00138-012-0450-4. 115, 119, 126

[143] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36 Vol.3, Aug 2004. doi: 10.1109/ICPR.2004.1334462. 115

[144] Chris Whiten, Robert Laganiere, and Guillaume-Alexandre Bilodeau. Efficient action recognition with mofreak. In *Computer and Robot Vision (CRV), 2013 International Conference on*, pages 319–325. IEEE, 2013. 117

[145] Kwan-Yee Kenneth Wong and R. Cipolla. Extracting spatiotemporal interest points using global information. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct 2007. doi: 10.1109/ICCV.2007.4408923. 121

[146] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct 2007. doi: 10.1109/ICCV.2007.4408988. 121

[147] Zhenzhong Lan, Ming Lin, Xuanchong Li, A. G. Hauptmann, and B. Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 204–212, June 2015. doi: 10.1109/CVPR.2015.7298616. 122, 126, 127, 128

[148] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. doi: 10.1109/CVPR.2007.383266. 123, 124

[149] F. Perronnin and D. Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3743–3752, June 2015. doi: 10.1109/CVPR.2015.7298998. 123, 139

[150] Jorge Sanchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013. ISSN 1573-1405. doi: 10.1007/s11263-013-0636-x. URL http://dx.doi.org/10.1007/s11263-013-0636-x. 123

[151] Florent Perronnin, Jorge Sanchez, and Thomas Mensink. *Improving the Fisher Kernel for Large-Scale Image Classification*, pages 143–156. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-15561-1. doi: 10.1007/978-3-642-15561-1_11. URL http://dx.doi.org/10.1007/978-3-642-15561-1_11. 123

[152] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3384–3391, June 2010. doi: 10.1109/CVPR.2010.5540009. 123, 126

[153] H. Jegou, F. Perronnin, M. Douze, J. SÃąnchez, P. PÃľrez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, Sept 2012. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.235. 123, 135

[154] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, chapter Action Recognition with Stacked Fisher Vectors,

pages 581–595. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10602-1. doi: 10.1007/978-3-319-10602-1_38. URL http://dx.doi.org/10.1007/978-3-319-10602-1_38. 123

[155] Y. Uchida and S. Sakazawa. Image retrieval with fisher vectors of binary features. In *2013 2nd IAPR Asian Conference on Pattern Recognition*, pages 23–28, Nov 2013. doi: 10.1109/ACPR.2013.6. 123, 124

[156] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *In Advances in Neural Information Processing Systems 11*, pages 487–493. MIT Press, 1998. 125

[157] Vijay Chandrasekhar, Jie Lin, Olivier Morère, Hanlin Goh, and Antoine Veillard. A practical guide to cnns and fisher vectors for image instance retrieval. *Signal Processing*, 128:426–439, 2016. 126

[158] V. Kantorov and I. Laptev. Efficient feature extraction, encoding, and classification for action recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2593–2600, June 2014. doi: 10.1109/CVPR.2014.332.

[159] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109 – 125, 2016. ISSN 1077-3142. doi: http://dx.doi.org/10.1016/j.cviu.2016.03.013. URL http://www.sciencedirect.com/science/article/pii/S1077314216300091. 126

[160] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time action recognition with enhanced motion vector cnns. *arXiv preprint arXiv:1604.07669*, 2016. 127, 128

[161] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, June 2014. doi: 10.1109/CVPR.2014.223. 127, 128

[162] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 127, 128